Essays on the use of e-Learning in Statistics and the Implementation of Statistical Software

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum politicarum
(Doktor der Wirtschaftswissenschaft)
eingereicht an der
Wirtschaftswissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Dipl.-Kfm. Uwe Ziegenhagen M.Sc.
Nauen, 02.09.1977

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Wirtschaftswissenschaftlichen Fakultät:
Prof. Dr. Oliver Günther

Gutachter:
1. Prof. Dr. Wolfgang Härdle
2. Prof. Dr. Ostap Okhrin

eingereicht am: 29.09.2008
Tag des Kolloqiums: 30.01.2009

Acknowledgment

I would like to express my deep gratitude to Prof. Dr. Wolfgang Härdle for constant support througout the years I have spent in his institute. Without his commitment and helpful comments, this work would not exist in its present form.

I gratefully acknowledge the financial support of Deutsche Forschungsgemeinschaft through the Collaborative Research Center 649 "Economic Risk" and the Society for Economics and Management at Humboldt-Universität zu Berlin.

Finally I would like to thank my colleagues from the Institute of Statistics and Econometrics as well as my parents and friends who provided help whenever I needed some.

Abstract

The following doctoral thesis collects the papers the author has written with his coauthors on e-Learning and statistical software. The chapters 2 to 5 are devoted to selected aspects of e-Learning, the chapters 6 to 9 describe the development of the statistical programming environment Yxilon. In chapter 2, coauthored by Wolfgang Härdle and Sigbert Klinke, the question whether and how computational elements should be integrated into the canon of methodological education and where e-techniques have their limits in statistics education is discussed. Chapter 3, coauthored by Wolfgang Härdle and Sigbert Klinke, gives reviews of different e-learning platforms for statistics and reveals facts that may be taken into account for future e-learning platforms in statistics and related fields. Chapter 4, written with Wolfgang Härdle and Sigbert Klinke, discusses two papers published in International Statistical Review which both offer a technical solution to improve the understanding of statistics by students. Chapter 5, coauthored by Wolfgang Härdle and Sigbert Klinke, describes web-related techniques for teaching statistics. It furthermore introduces the Quantnet platform, a framework to manage scientific code and data. In chapter 6, coauthored by Wolfgang Härdle and Sigbert Klinke, the requirements for a statistical engine are discussed. Chapter 7, written jointly with Yuval Guri and Sigbert Klinke, explains ideas which led to the reimplementation of the XploRe language. In chapter 8, coauthored by Wolfgang Härdle and Sigbert Klinke, the implemented client/server structure of the Yxilon platform is laid out in terms of technical features. The server and the communication protocol are described together with the developed Java client featuring the Jasplot graphics engine. Finally chapter 9 describes the structure of the Yxilon environment in its present form.

Zusammenfassung

Die vorliegende Doktorarbeit bündelt die Veröffentlichungen des Autors und seiner Koautoren zu den Themen e-Learning und statistischer Software. Die Kapitel 2 bis 5 sind Aspekten des e-Learning gewidmet, die Kapitel 6 bis 9 beschreiben die Entwicklung der statistischen Programmiersprache Yxilon. In Kapitel 2, Koautoren Wolfgang Härdle und Sigbert Klinke, wird erörtert, ob und wie computerbasierte Elemente in den Kanon der methodischen Bildung integriert werden sollen und wo die Grenzen des e-Learning in der Statistik-Ausbildung liegen. Kapitel 3, Koautoren Wolfgang Härdle und Sigbert Klinke, gibt Einschätzungen verschiedener e-Learning Plattformen und beschreibt Punkte, die bei der Entwicklung von e-Learning Plattformen berücksichtigt werden sollten. Kapitel 4, geschrieben mit Wolfgang Härdle und Sigbert Klinke, diskutiert zwei Veröffentlichungen in der International Statistical Review, die eine technische Lösung für die Verbesserung des Verständnisses der Statistik-Lehre vorstellen. Kapitel 5, Koautoren Wolfgang Härdle und Sigbert Klinke, beschreibt die Anwendung von Web-Techniken für die Lehre in Statistik. Weiterhin stellt es die Quantnet Plattform vor, eine Plattform für die Verwaltung von Programmen und Daten. In Kapitel 6, Koautoren Wolfgang Härdle und Sigbert Klinke, diskutieren die Autoren die Anforderungen an eine Statistical Engine. Kapitel 7, geschrieben mit Yuval Guri und Sigbert Klinke, erläutert die Ideen, die zur Re-Implementierung der XploRe Sprache geführt haben und diskutiert ausgewählte technische Aspekte der Yxilon Plattform wie Objektdatenbank und die Erzeugung von kompilierbarem Code für Hochsprachen. In Kapitel 8, Koautoren Wolfgang Härdle und Sigbert Klinke, wird die implementierte Client-Server Struktur beschrieben. Server und Kommunikationsprotokoll werden zusammen mit dem entwickelten Client und der Grafik-Engine beschrieben. Das letzte Kapitel, beschreibt die Struktur der Yxilon Plattform in ihrer jetzigen Form.

Inhaltsverzeichnis

1 Introduction

Without any doubt computers have changed our lives: the way we work, learn or process information. For a majority of employees computers are inseparable part of their work and private life as, according to a recent study (Bitkom, 2008) about two third of all German employees use a computer for their work.

For statistics as a discipline the use of computers have open a door to new levels of data analysis. Before computers were introduced data analysis was a tedious procedure, as it involved a lot of manual work and huge amounts of paper or punchcards (Eckert and McPherson, 1984). The use of computers not only allowed the use of large datasets – although even the definition of a 'large dataset' is subject to constant change – but also the implementation of methods such as bootstrapping or explorative data analysis (EDA) which seem impossible without a computer.

With Yxilon, we tried to develop a successor of the statistical programming language XploRe, allowing applied data analysis using a modern language-based approach. Based on common principles and new ideas this software was intended to be used in research and education. Although the final goal, the completion of the Yxilon framework, could not achieved, the project gave important insights into statistics, software development and impulses for other software projects.
With Yxilon we also wanted to enrich the process of learning statistics. Statistics as a discipline joins various topics such as mathematics, politics, economics and computer science. Experience has shown that a significant proportion of students has problems in successfully applying their knowledge about economics in a statistical context. E-learning environments can be of use if the successfully map real-world situation to statistical contexts.

The following doctoral thesis collects the papers the author has written on e-Learning and statistical software with his coauthors. The chapters 2 to 5 are devoted to selected aspects of e-Learning, the chapters 6 to 9 describe the development of the statistical programming environment Yxilon.

In chapter 2, coauthored by Wolfgang Härdle and Sigbert Klinke, the necessity of practical, computer-based data analysis in statistics curricula is discussed together with the question whether and how computational elements should be integrated into the canon of methodological education and where e-techniques have their limits in statistics education. It is tried to answer the question if students should see and study high-level programming code right at the beginning of their studies or if this may be asked too

much of them. We give recommendations which technologies can be presented during classes and which computational elements can reoccur at increasing level of complexity during different courses.

Chapter 3, which was coauthored by Wolfgang Härdle and Sigbert Klinke, gives personal reviews of different e-learning platforms for statistics and reveals facts that may be taken into account for future e-learning platforms in statistics and related fields. One of the most striking discoveries of the analysis is that students of statistics actually do not use electronic media in the desired frequency and actually rely more on print media such as books, copies of slides.

Chapter 4, written jointly with Wolfgang Härdle and Sigbert Klinke, discusses two papers published in International Statistical Review, namely Darius et al. (2007) and Nolan and Lang (2007) who both offer a technical solution to improve the understanding of statistics by students. Nolan and Lang (2007) propose a document system whose aim is to allow students follow the decision making process of a statistical analyst; Darius et al. (2007) present a set of Java applets to assist the students´ understanding of experimental design. As there has been invented a plethora of interactive tools with the aim to help students to receive a better understanding of statistics and to develop a ßtatistical thinking“ of which only a few have been successful the following questions rise: Why do (economics) students have those severe problems with introductory statistics that additional tools are required? Which impact do e-learning tools have on our teaching? What can we expect to happen in the future?

Chapter 5, coauthored by Wolfgang Härdle and Sigbert Klinke, describes the application of web-related techniques for teaching statistics. It describes various applications of wikis, shows how the learning content management system Moodle can be used to evaluate students' knowledge and describes our experience with the use of videos in teaching. It furthermore introduces the Quantnet platform, a framework to manage scientific code and data.

In chapter 6, coauthored by Wolfgang Härdle and Sigbert Klinke, we discuss the requirements for a statistical engine. With spreadsheet software being published annually with more and more features and the decline of commercial packages such as S-Plus certain requirements concerning statistical computer power, usability and extensibility can be defined. The chapter furthermore introduces Yxilon as a vertically integrable statistical software package.

Chapter 7, written jointly with Yuval Guri and Sigbert Klinke, explains the ideas which have led to the reimplementation of the XploRe language and discusses technical aspects of the Yxilon platform such as the object database and the originally planned generation of compilable code for high-level programming languages such as Java and C++.

In chapter 8, coauthored by Wolfgang Härdle and Sigbert Klinke, the implemented cli-

ent/server structure of the Yxilon platform is laid out in terms of technical features. The server and the communication protocol are described together with the developed Java client featuring the Jasplot package by Yamamoto et al. (2007) as used graphics engine.

Finally chapter 9 describes the structure of the Yxilon environment in its present form, with focus to the client/server architecture and the implementation of the graphics architecture that was introduced in chapter 8.

2 Integrable e-lements for Statistics Education (JSCS 2005)

Abstract

Without doubt modern education in statistics must involve practical, computer-based data analysis but the question arises whether and how computational elements should be integrated into the canon of methodological education. Should the student see and study high-level programming code right at the beginning of his or her studies? Which technology can be presented during class and which computational elements can re-occur (at increasing level of complexity) during the different courses?

In this paper we address these questions and discuss where *e*-techniques have their limits in statistics education.

Keywords electronic books, hypertext, e-supported teaching, statistical software

JEL codes I21, C19,

2.1 Introduction

Since the entering of modern computing equipment into schools and universities there have been increased efforts to use computers not only for research and numerical computations but also for the education of students. While traditional textbooks on statistics are usually restricted to small examples, computers offer great opportunities to enrich the teaching of statistics by the means of explaining animations or on-the-fly computations of large real-world datasets.

But each new technology does not only hold opportunities and advantages there may also be hazards or risks. And a sentence stated by John Tukey in 1965 should be taken into consideration: Ëach new generation of computers offers us new possiblities, at a time when we are far from using most of the possibilities offered by those already obsolete."

We will outline our thoughts about the integration of electronic and computational elements into statistics along the courses tought at the School for Business and Economics of Humboldt-Universität zu Berlin.

Students following different course programs are taught at the Institute for Statistics and Econometrics at Humboldt-Universität zu Berlin, German Business Administrati-

on and Economics students as well as students from international bachelor or master classes or the math department.

After finishing the bachelor level, where statistics is taught in a two-term lecture three hours a week, the students who choose statistics as one main subject are required to take a course on multivariate data analysis, that enables them to understand the basics of probability theory and to analyze high-dimensional data by the means of cluster-, principal component- and factor analysis.

On the basis of this lecture the student has further options: While *Non- and Semiparametric Modelling* focuses on nonparametric density estimation and regression the student learns in the computational statistics courses the applied data analysis with SPSS and real world data. Supplementary to the *Computerbased Statistics* courses are the *XploRe Introductory Course* and the *Numerics Introductory Course* focussing on the practical work with XploRe respectively the numerical details of selected algorithms.

Statistics of financial markets is the third specialisation taught at Humboldt-Universität, covered are theoretical and practical aspects of option pricing, risk-management and time series modelling.

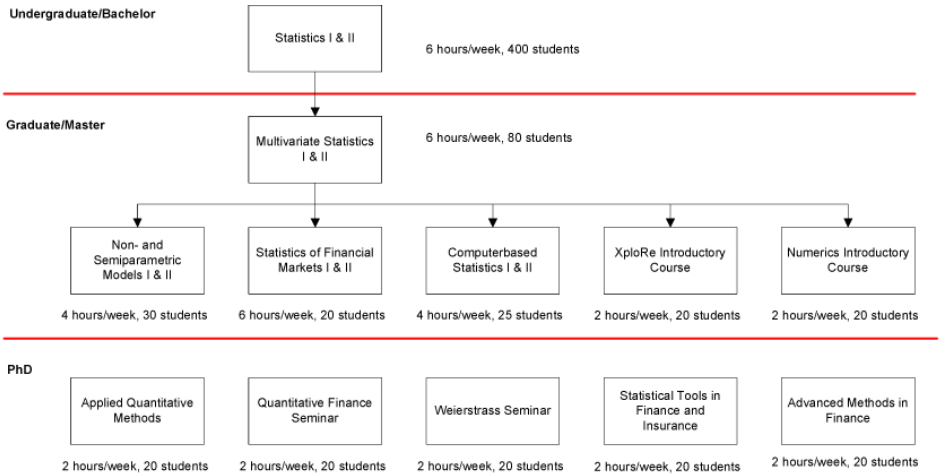


Abbildung 2.1: Typical one year cycle of statistics courses in Berlin

At the master level there are currently five courses also aimed at PhD students: In *Applied Quantitative Methods* and the *Weierstrass Seminar* recent developments of mathematical statistics and data analysis are discussed, *Advanced Methods in Finance*, the *Quantitative Finance Seminar* and *Statistical Tools in Finance and Insurance* deal with the field of financial mathematics.

2.2 Traditional and Modern Teaching Material

The main components of MM*Stat (Müller et al., 2000), that was developed to support the undergraduate teaching of statistics, are *lecture units*, *additional information* and *(non)interactive examples*, presented in a style similar to filing cards. This structure, called *MD*Booklet*, has also been used for *Numerical Methods in Statistics* and *Finance Introductory Course* (http://www.quantlet.com/mdstat/products.html). The lecture units con-

tain common and well-known topics as basic concepts of statistics, basics of probability calculus and sampling theory.

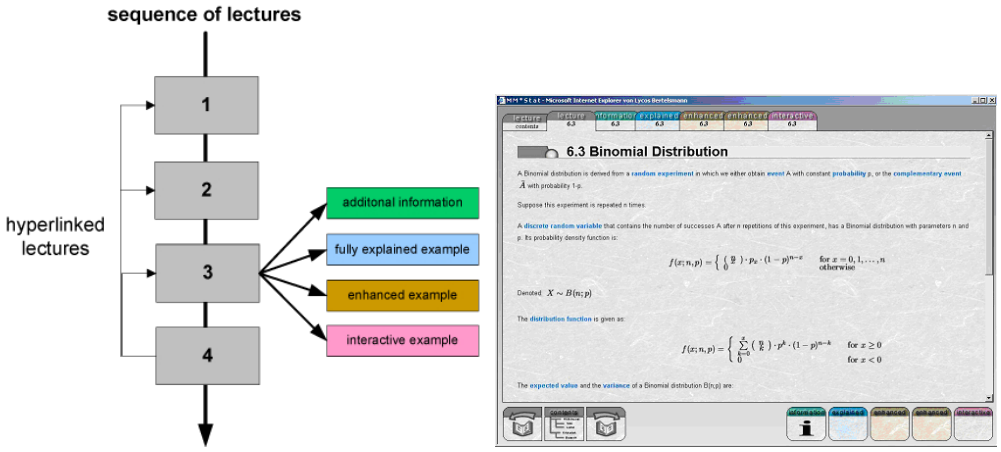


Abbildung 2.2: MM*Stat: Layer Architecture and Screenshot

Each "lecture"filing card provides the basic concepts of the methods considered together with definitions, important formulas and graphics. "Fully explained", Ënhancedänd Ïnteractiveëxamples show how these formulas can be applied. The first explain standard classbook examples in detail while the latter discuss important aspects of more advanced examples. The interactive examples are one of the main features of MM*Stat. By using a combination of embedded Java-based *XploRe Quantlet Client* (Borak, S., Härdle, W., Lehmann, H., 2005) and a remote or locally installed *XploRe Quantlet Server* the student can compute distribution functions, histograms and test results. Compared with classical textbooks where effects of parameter changes can only be presented as sequence of graphics or tables, the user can experience the results of different settings in realtime.

The repetition of concepts and methods introduced earlier has been implemented in two different ways: To see or recall the definition of a used term, a glossary has been written and important words within the different lectures inside MM*Stat are linked to this glossary. Furthermore the student can complete multiple choice questions by selecting radio buttons, the results are given instantly.

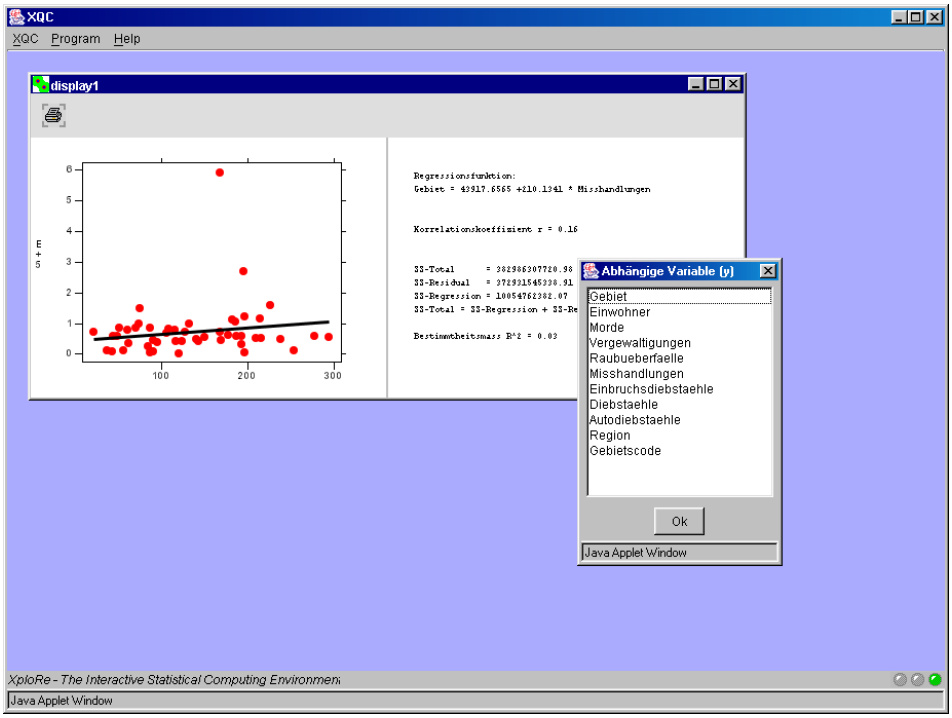


Abbildung 2.3: Example for linear regression in MM*Stat

The target group of MM*Stat, the undergraduate students, have different levels of literacy concerning computers, while some are firm in common office software tools others have little experience. So it was decided to hide the source code of the interactive examples from the user, to lower the inhibition threshold of using the software. Figure 2.3 shows a typical example for linear regression, which allows the user to choose the dependent and independent variable and returns the regression plot and the parameters of the model.

The combination of XploRe Quantlet Server and Client has not only been used in MM*Stat, furthermore it is used in the DoSS@d system (Mori, Y., Yamamoto, Y. and Yadohisa, H., 2003) and the books published by the members of the authors' institute (http://www.xplore-stat.de/ebooks/ebooks.html) also use this framework, although in a different way.

As mentioned the source code for the interactive examples was hidden from the user in MM*Stat, from the target group of the advanced books, graduate and PhD students on the one, researchers and professional on the other hand, we assume a certain level of computational literacy. In each book respectively the corresponding slides selected pages contain links to HTML-pages that contain the source code of the example as well as links to two different implementations of this example.

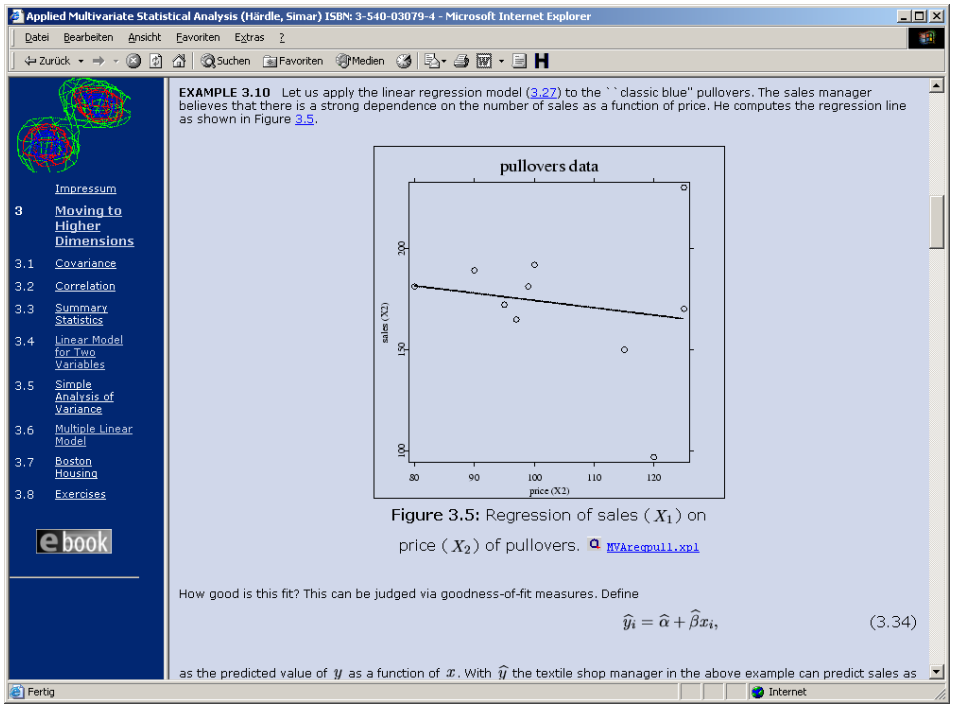


Abbildung 2.4: Applied Multivariate Statistics: HTML page with link to an example

These two implementations differ in that sense that the 'edit' page calls the XploRe Quantlet Client in editor mode, which means that the user can manipulate the source code of the example. From MM*Stat they vary from the textual component. While at undergraduate level regression is taught using a notation with sums, the more abstract matrix-based approach is used for graduate students.

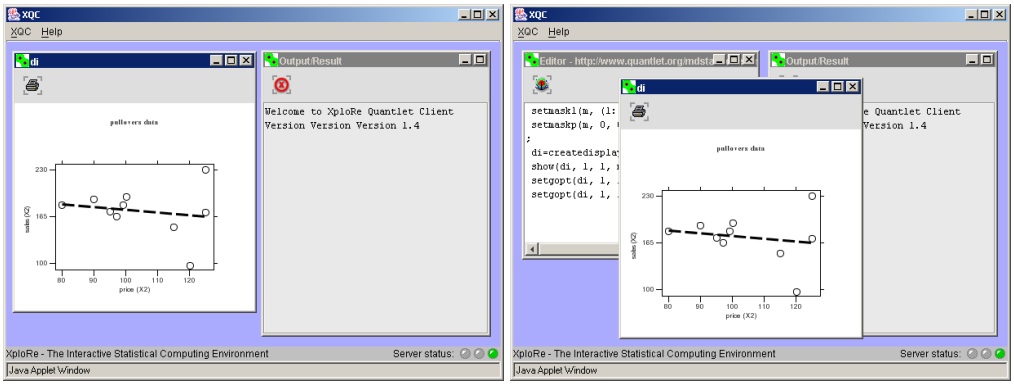


Abbildung 2.5: Applied Multivariate Statistics: *execute* and *edit* versions of an example

Figure 2.5 depicts two screenshort from the book *Applied Multivariate Analysis*, (Härdle and Simar, 2003), on the left hand side the 'run' version of the quantlet, on the right the editable 'edit' version.

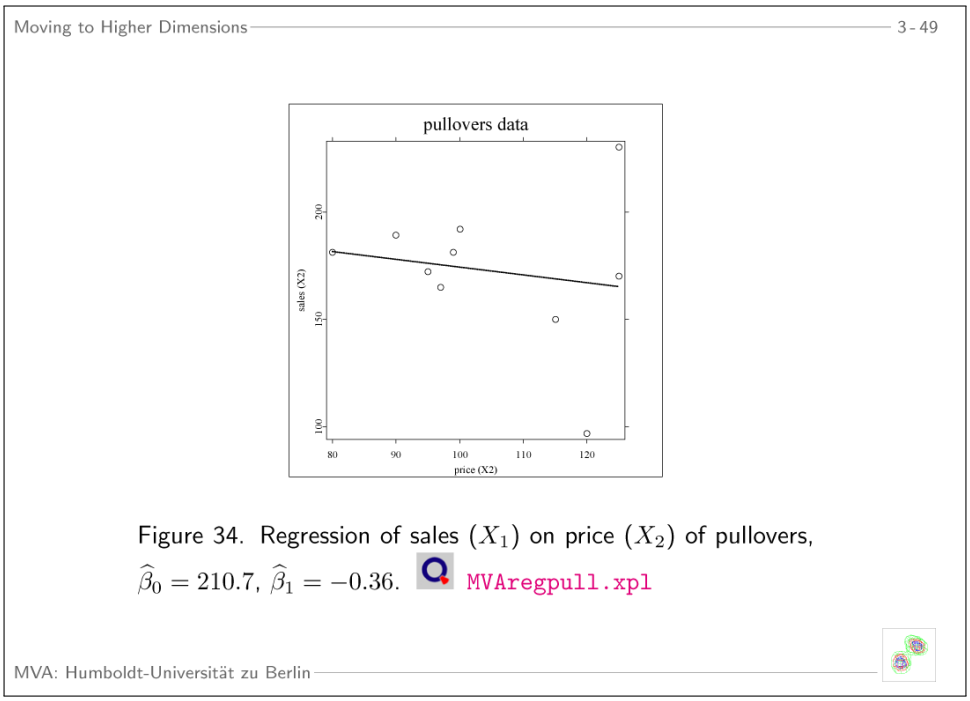


Abbildung 2.6: Applied Multivariate Statistics: slide with link to an example

Abbildung 2.7: Applied Multivariate Statistics: homepage of the linked example

2.3 XploRe, Xplore Quantlet Server and Yxilon

Besides SPSS which is used for the computational statistics courses the software package XploRe is used for educating students at Humboldt-Universität. Developed in cooperation with MD*Tech (http://www.mdtech.de), XploRe is a full-featured statistical programming language. Using a matrix-oriented approach in combination with a C-style syntax a great variety of diverse statistical problems can be handled conveniently.

The implementation of XploRe incorporates ideas from the requirements of statistical software mentioned by Chambers and Lang (1999): usage from multiple front-ends, extensibility on language and native core level, interactive graphics and the inclusion of existing code (`C`, `Fortran`).

With Yxilon (Guri et al., 2005) MD*Tech and Humboldt-Universität zu Berlin agreed on a new research project with the aim to supply a freely available statistical environment and to allow the implementation of recent developments in software technology while remaining fully compatible with XploRe.

The most dramatic change compared with XploRe or other packages as *R* or *Jasp* (Fujiwara et al., 2000) is the way the code is executed: Besides the interpretation by a so called runtime environment the idea is to compile directly to C++ and Java. Further changes are:

- published under Free-BSD license
- fully independent from machine architecture by using platform-independent protocols and software frameworks
- improved modularity to allow convenient exchange of single components
- integration of dynamic data sources as databases and webservices
- increased computing performance simplifying the language definitions
- improved integration into standard office and business software

The software architecture of Yxilon is shown in Figure 2.8. Via a graphical or non-graphical user interface the user accesses the system. All objects and information are held in the object database, that communicates with the other parts as the parser or runtime environment. The parser takes the sourcecode provided by the user and converts them to either Java/C++ code or a binary format directly readable by the runtime engine. The advantage of source conversion is that the high-level, interpreted XploRe code, which is slower than binary code, can be compiled by the Java or C++ compiler to fast running machine code. Especially for computing-intensive applications as bootstrapping and simulation we expect significant improvements concerning execution time.

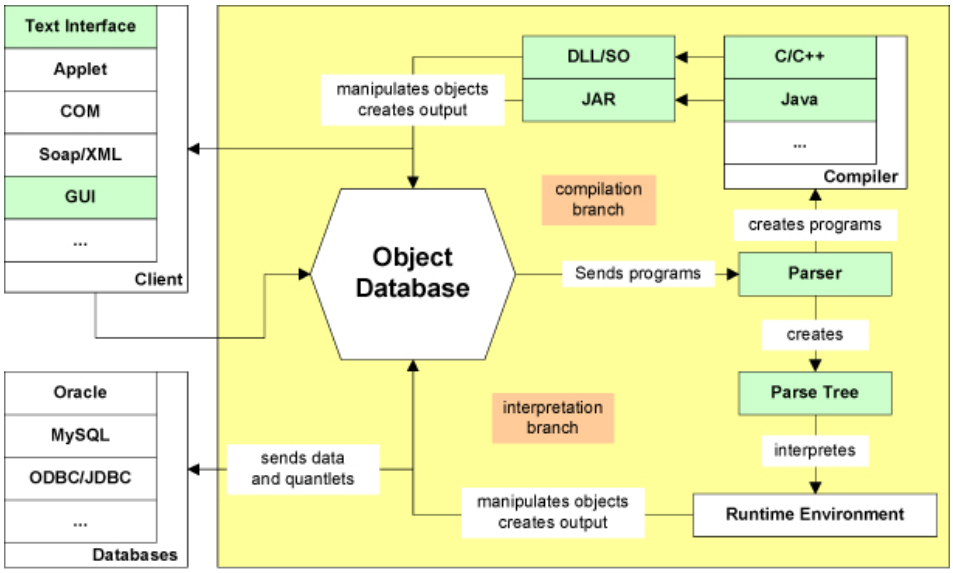


Abbildung 2.8: Architecture of the Yxilon Framework (green components in development)

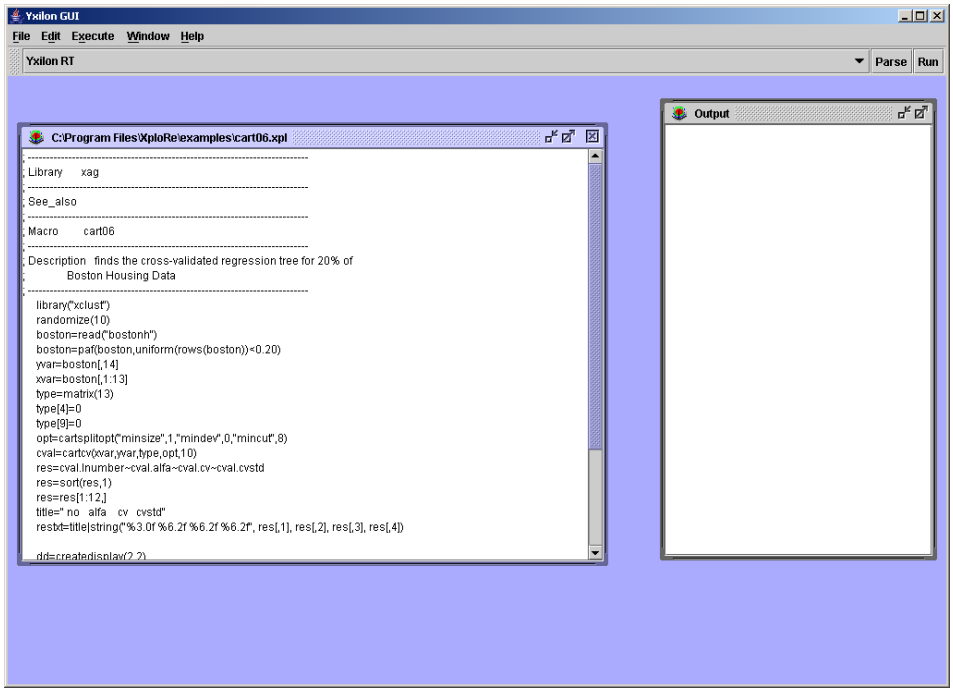


Abbildung 2.9: Screenshot of the Yxilon graphical user interface

2.4 Limits of *e*-lements in statistics education

Electronic and computational elements in education can be limited from a variety of different reasons as the technical implementation or the educational approach.

MM*Stat relies on a complex framework of HTML, Cascading Style Sheets, Java and JavaScript. While HTML and Java were already well-established standards, JavaScript and Cascading Style Sheets were implemented in different ways for each browser, so extensive work for each major webbrowser had to be invested.

From the educational point of view each use of electronic media should be questioned carefully. An analysis by Brandes (2004) revealed no significant improvements in the results of those students who used MM*Stat in comparison to those who used classical textbooks. The educational limits also lie in the way the information is presented. The classical way of using a textbook or blackboards also offers the opportunity to include on-the-fly notes from either the teacher or the student. Electronic teaching solutions either do not offer this capability at all or require significant technical ressources.

3 e-Learning Statistics – A Selective Review (Compstat 2006)

Modern computing equipment is present at schools and universities at all levels of education.[1] In the statistical sciences computers offer great opportunities to enrich the learning process by the means of e.g. animations, software integration or on-the-fly computations.

A personal review of different e-learning platforms for statistics is done in this paper. This review reveals facts that could be taken into account for future e-learning platforms in statistics. One of the most striking discoveries of our analysis is that students of statistics actually do *not* use electronic media in the desired frequency and actually rely more on print media such as books, copies of slides, etc.

Keywords e-learning, electronic books, hypertext courseware, statistical software
AMS Classification 97U70
JEL Classification I21, C19

3.1 Introduction

There has been a plethora of approaches to find the 'holy grail' of e-learning statistics, satisfying the needs of teachers and students as well. In fact the demand of students and teachers for e-learning must be correctly balanced as the following example illustrates.

The aim of a student, for example, is to get immediate and fast access to media which aid him or her in the acquisition of knowledge to pass the exam. The teacher on the other hand is interested in improving his classes and providing the student with knowledge for sophisticated data analyses under different conditions. These aims are generally conflicting and hardly mappable into a single e-learning system.

In addition each course level requires its special e-learning architecture since the degree of computer literacy and the willingness to perform interactive statistical learning tasks is, of course, a function of this level. At ISE/CASE the education is based on the following elements: introductory statistics and probability theory in the first year, multivariate statistics and first computational steps in the second year and specialisation in a theoretical or applied field (e.g. finance, insurance) and research seminars in the third year. An overview about the course structure is given in Table 3.1.

[1]This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 'Economic Risk'.

Tabelle 3.1: Overview of the course structure at ISE

Degree	Group	Class	e-Media
Bachelor	Introduction	Statistics I & II	MM*Stat Q & A
	Multivariate Statistics	Multivariate Statistics I	e-stat
		XploRe Introductory Course	XploRe
	Applied Statistics	Computerbased Statistics I & II	eBooks
		Data Mining/Statistical Learning	Excel
		Numerical Introductory Course	
	Privatissimum	Privatissimum	
		Bachelor Thesis	
Master	Multivariate Statistics	Multivariate Statistics I	e-stat
		XploRe Introductory Course	XploRe
	Statistics of Fin. Markets	Statistics of Fin. Markets I & II	R
	Advanced Statistics	Multivariate Statistics II	Matlab
		Non- and Semiparametrics I & II	MD*Booklets
		Numerical Introductory Course	
		Applied Quantitative Methods	
	Privatissimum	Privatissimum	
		Master Thesis	
PhD	Financial Statistics	Quantitative Finance Seminar	XploRe
		Adv. Stat. Methods in Finance	eBooks
		Mathematical Statistics Seminar	
		Statistical Tools in Finance and Insurance	

During the last ten years we have developed several systems on our own, in addition we have been involved in a variety of e-learning projects. Based on our experience with these projects we observed that:

1. Interactivity is not appreciated or understood.
2. HTML pages and screenshots of interactive examples are often printed out for take home studies.
3. Discussion groups are rarely used.
4. Feedback through uploading of e.g. homework is often refused.

This might all be a consequence of technical issues (internet browser, operating system), we do believe though that e-learning in statistics should not invest too much optimism but rather provide a thoughtful set of really integrable e-learning examples.

This paper is organized as follows. In Section 2 we describe a selection of e-learning systems, in Section 3 we report our main findings. Section 4 concludes the findings with a recommendation on future e-learning architectures.

3.2 Modern e-learning materials

3.2.1 MM*Stat

A first-year student learning statistics needs to be introduced into descriptive statistics and the basic concepts of probability theory, statistical testing and linear regression. The highschool training puts him on a level of knowledge which typically incorporates the judgement of roulette or lottery outcomes but not the testing of hypotheses and the graphical display of data. The idea of MM*Stat Müller et al. (2000), available online at `http://www.quantlet.com/mdstat/mmstat.html`, was to start from this level of knowledge and to introduce the students into basic concepts of graphical data analysis.

Figure 3.1 shows a screenshot for the English edition of MM*Stat and depicts the HTML-based filing card structure: Each 'lecture' filing card provides basic concepts of methods considered together with definitions, important formulas and graphics. 'Fully explained', 'Enhanced' and 'Interactive' examples are linked to the lecture card and show how the formulas can be applied. This *MD*Booklet* structure was implemented manually for the German edition, for other languages (see Table 3.2) an automatic conversion utility Klinke and Witzel (2002) had been developed on the basis of LaTeX2HTML.

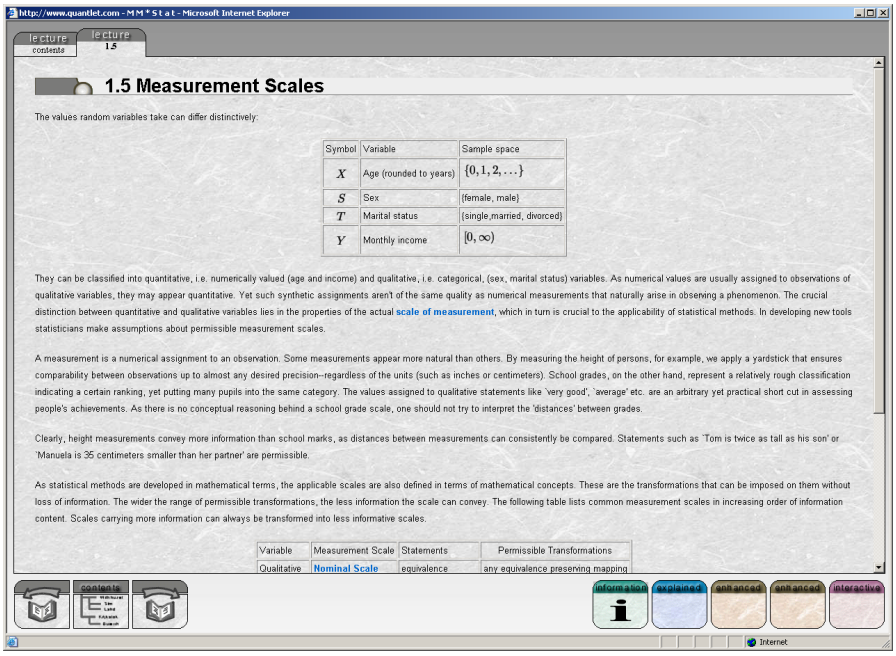


Abbildung 3.1: Screenshot of MM*Stat, English version

Tabelle 3.2: Languages covered by MM*Stat

Arab (in development)	Czech
Dutch	English
French	German
Italian	Portuguese
Slovenian	Spanish
Turkish	

For the integration of interactive examples a combination of an embedded XploRe Quantlet Client Borak et al. (2005) and a local or remote XploRe Quantlet Server (XQS) is used. Thus MM*Stat enables the student to compute distribution functions, draw histograms or derive results for statistical tests in the web browser.

In our opinion MM*Stat provides certain advantages compared with textbooks. On the one hand printed books are usually restricted to small datasets since it is exhausting to compute e.g. a regression line for a dataset of 100 value pairs, on the other hand the effects of of a parameter change can only be depicted as a sequence of graphs and tables. With MM*Stat, real-world datasets can be used, effects of parameter changes are shown in real-time.

One of the cornerstones of teaching, the repetition of earlier introduced concepts and methods is implemented in two different ways: to see or recall the definition of a used term, a glossary has been written, important words within the different lectures are linked to this glossary. In addition each chapter contains multiple choice questions, via JavaScript these questions are evaluated on-the-fly, the results are shown to the student.

3.2.2 Electronic Books

To generate an added value for print media the XQS technology used for MM*Stat has been successfully applied to books, which have been designed for 2nd year and above students. In the current implementation each book (see `http://www.xplore-stat.de/ebooks/ebooks.html`), its HTML version and the corresponding set of slides contain references to web pages on a webserver. These HTML pages show the XploRe source code and allow to run the example in a Java applet, a screenshot of 'edit' version of a quantlet, taken from the book *Applied Multivariate Analysis* Härdle and Simar (2003), is given in Figure 3.2.

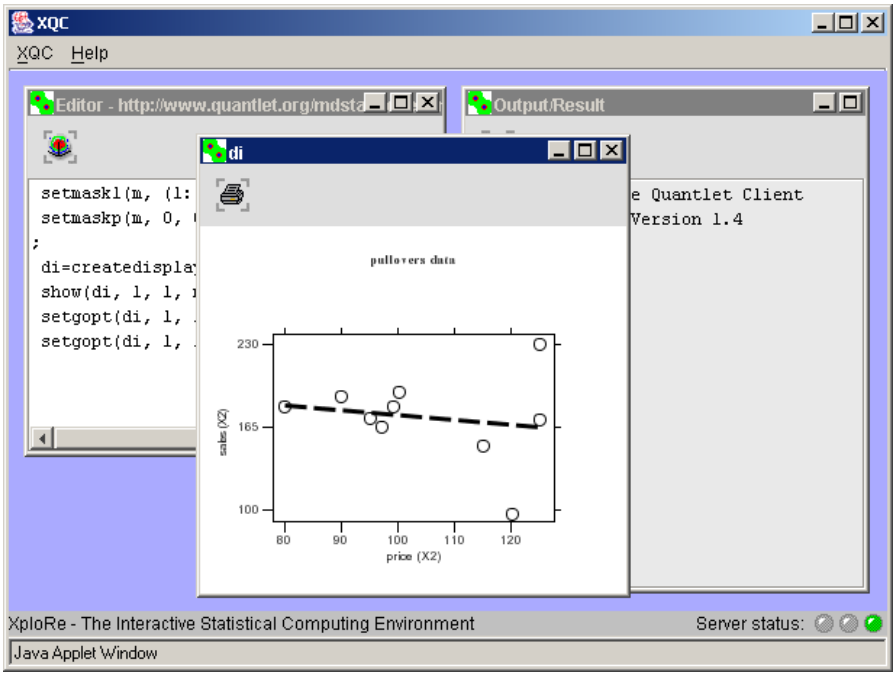


Abbildung 3.2: Screenshot of an interactive example from the 'Applied Multivariate Statistics' book

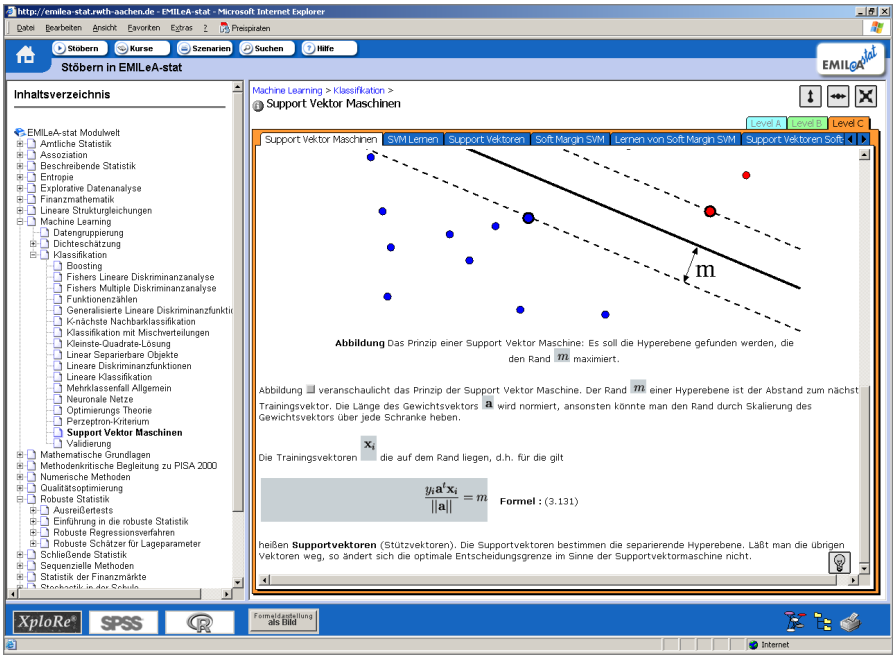


Abbildung 3.3: Screenshot of the e-stat user interface: navigation toolbar and content page

3.2.3 e-stat

The aim of the e-stat project (`http://emilea-stat.rwth-aachen.de/`) was to deliver a significant contribution for the improvement of the practical statistical education. The content modules were provided as XML by the project partners, a consortium of nine German universities. The XML code was then equipped with dedicated views and scenarios. The targeted user groups were not only students but also people with a general interest in statistics.

The various views and scenarios, which are linked to real-world applications include Cramer et al. (2002):

- Method-based: learning of methods along a predefined way of modules with terms, examples and exercises
- Problem-based: based on a simplified description of the problem, a consulting component then proposes a suitable solution
- View-based: problem description from specific areas serve as examples to deal with the underlying terms

To provide suitable information for different groups of users the e-stat system offers three levels (introductory, applied, advanced) of abstraction with increasing degrees of difficulty. Figure 3.3 shows a screenshot of the e-stat web site: On the left there is a list of all available modules which are then displayed in the right frame.

3.2.4 Q&A

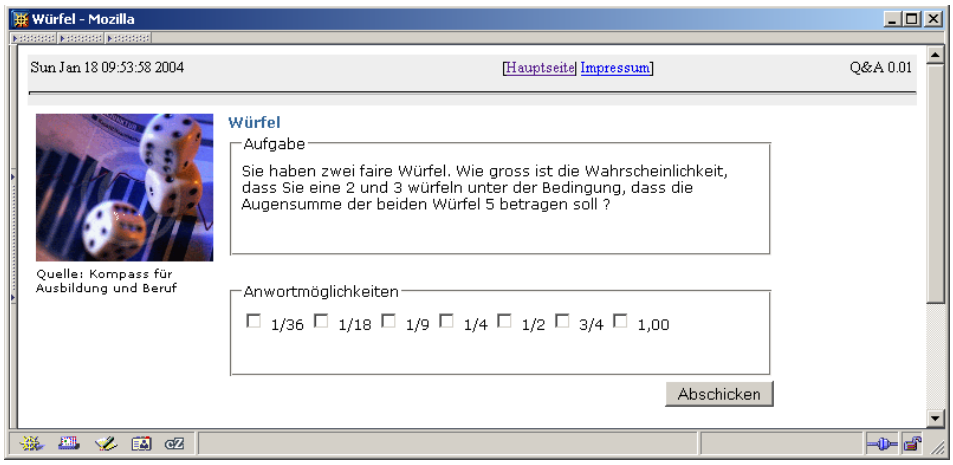


Abbildung 3.4: A 'simple' Q&A exercise: Throwing two dices.

The Q&A system (Klinke, 2004) provides an interactive environment for the exam preparation in undergraduate courses of statistics. The platform handles two types of exercises: 'simple' and 'variable' exercises. For both types of exercises a problem is presented to the student with multiple choice answers, see Figure 3.4). The homepage of the project is `http://stirner.wiwi.hu-berlin.de/qa/`.

Q&A allows to check for common student mistakes. Let us consider for example a simple exercise, shown in Figure 3.4. Assume two dices are thrown. What is the probability that one dice shows '2' and the other '3' under the condition that the sum of both dices is '5'?

Typically the students make two mistakes. They do not recognize that they have a) to compute a conditional probability and b) take into account that there are two elementary events.

Depending on the answer the students select, we can determine their most common mistake. We categorized the answers of 215 students into seven classes:

1. a) wrong, b) wrong 23%,
2. a) wrong, b) correct 39%,
3. a) correct, b) wrong 6%,
4. a) correct, b) correct 16%,
5. not answered at all/other answers 11%,
6. an answer, which belongs to the other version 5%,

As a reply to the first four groups of errors different web pages with hints on the correct solution are given.

3.2.5 Moodle

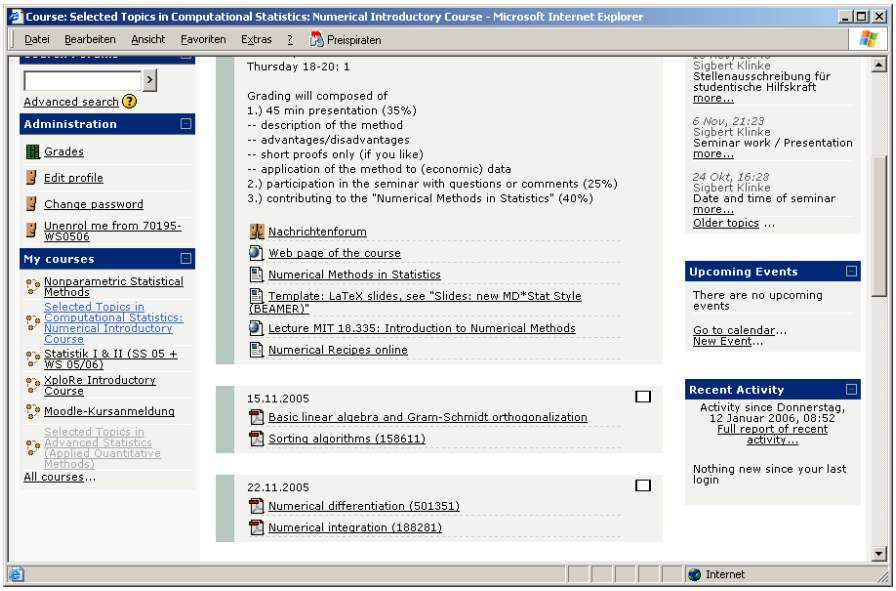


Abbildung 3.5: The 'Numerical Introductory Course' in the Moodle system

Moodle (Kristöfl, 2005) is a course management system (CMS) - a free, Open Source software package designed using sound pedagogical principles, to help lecturers create effective online learning communities. Moodle has a large and diverse user community with over 75,000 registered users at `http://moodle.org`, covering 70 languages in 138 countries.

At Humboldt-Universität zu Berlin the Moodle system is maintained by the 'Multimedia Lern- and Lehrzentrum' from the computing center of the university. At ISE/-CASE the work with Moodle started at the beginning of 2005; since that time it has been used for several courses at Bachelor and Master level.
Moodle allows to

1. structure lectures according to a set of topics or a time table,
2. maintain a class forum and forums related to each topic or date of the time table,
3. group students to solve different tasks or exercises,
4. upload and manage documents for lecturers, e.g. lecture slides, and students, e.g. solutions for exercises and
5. generate online exercises with direct evaluation.

The Moodle system, a screenshot is depicted in Figure 3.5, provides many more features, yet only a few have been used by the ISE. The first question that each lecturer should answer is: Why should one of their student use such a system? Our answer is that we have moved the teaching materials, e.g. slides, information about the grading process from the web pages of the institute to the Moodle system. Thus to download the formulary, the exercise collections or the slides the students have to enter the Moodle system. To receive an account the students have to provide a valid e-mail address, this allows us, together with the email features of Moodle, to make important announcements to the whole class easily.

3.2.6 Other Packages

Although spreadsheet software such as Excel may not be the optimal choice for data analysis McCullough and Wilson (2005), they are suitable tools for the statistics education. They are available on almost every computer and the majority of the students knows how to operate them. At the ISE Excel sheets are mainly used in the education of undergraduate students, e.g. to visualize the Central Limit Theorem or parameter changes of distribution functions. The built-in functions are easily accessible, without knowing a high-level programming language such as Java or C++ formulas may be edited or graphical user interface components such as sliders, radiobuttons and pull-down menus be used. Figure 3.6 displays a screenshot of a spreadsheet to visualize eight different probability density functions, via sliders the parameters can be modified.

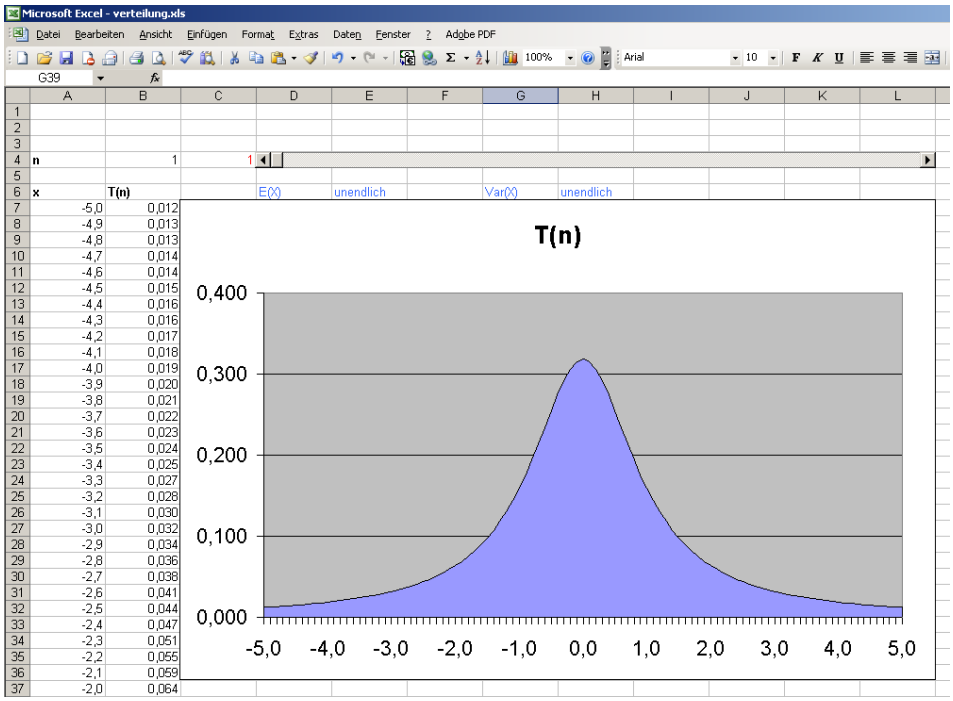


Abbildung 3.6: Visualisation of parameter changes with Excel

DoLStat@d Mori et al. (2003) has been developed at Okayama University, Japan by Yuichi Mori and colleagues. This web based learning system, available online at `http://mo161.soci.ous.ac.jp/@d/DoLStat/index.html`, provides various courses which are classified into major categories such as 'General statistics', 'Research field' and 'Statistical method'. Each course contains real world data with their analysis stories, see Figure 3.7, which are from a data-oriented statistical database system. The provided data sets are designed for the learning purpose of the course and are ordered in educational perspective.

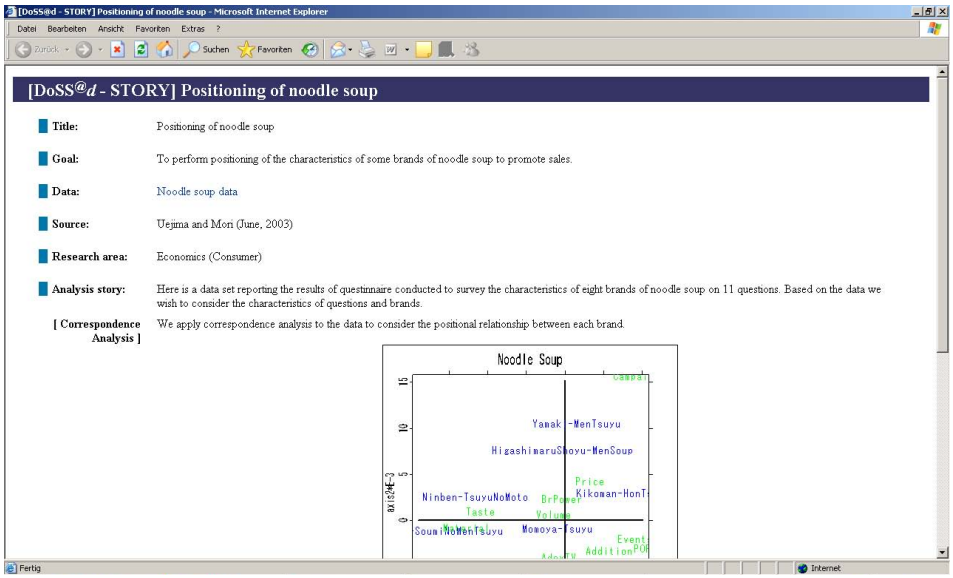


Abbildung 3.7: Screenshot of DoLStat@d, describing goals, data and story of an analysis

'Neue Statistik' (`http://www.neuestatistik.de/`) is a multimedial teaching platform for statistics for German universities. The aim Grune (2002) was to replace the formal and mathematical way in the education of undergraduate students by a problem-oriented and practical approach. This includes the usage of animations, diagrams and video sequences. Part of the system is the free teaching software 'Statistiklabor', which is based on *R*.

3.3 Evaluation

There are four different aspects in the work with e-learning environments, the technical, the content and the attitudes from lecturers and students.

For MM*Stat the issues arose mostly from the technical side. When MM*Stat was developed, the two leading web browsers were the partly incompatible Netscape 4 and Internet Explorer 5, therefore the work focused on these two programs. Incompatibilities with newer browsers such as Internet Explorer 6 and Mozilla Firebird prevented the successful usage of MM*Stat inside the lectures and exercise classes. For languages such as Arab the LaTeX/LaTeX2HTML software does not provide enough support, here a huge load of manual work is necessary.

A statistical analysis of questionnaires among students revealed that MM*Stat partially replaced traditional media such as books for a significant proportion of students but did not lead to better grades.

The electronic books are HTML versions of printed books, enriched with interactive examples. These examples require the Java Runtime Environment which is an additional barrier for the usage. There were no further adjustments paying tribute to special

requirements of electronic media. The quality of the content largely depends on the authors. Since the students prefer to have a printed copy of the book, the eBooks mainly serve as easily searchable encyclopedia.

The e-stat project mainly suffered from technical and content issues. The complicated structure of modules requires a certain amount of knowledge, especially of XML. Furthermore the different modules are diverse, in their quality and amount. For some modules there are extensive descriptions at all levels, for a large proportion of modules there are only information at one of three levels.

The basic idea of Q&A was that the student invests time on thinking about the correct answer for an exercise, however an analysis of the generated log files revealed an opposite behavior. Instead of carefully selecting an answer, the student usually click on the first solution, then the second,... until he or she has found the correct solution. Q&A also requires a significant amount of work from the lecturer. For each common error a web page with hints and comments has to be written.

Specialized systems such as Moodle may provide advantages especially in terms of options for the multiple choice questions. However, the use of a standard software limits the flexibility in comparison with self-developed solutions, e.g. in the design of the answer pages.

Moodle differs from the other environments since its main purpose is to manage contents for lectures and to provide forums, chats and content storage for various types of lectures. However the complexity of Moodle requires an initial investment from the lecturer as they are forced to structure their lectures, either on topics or a time schedule. Furthermore they have to investigate which of the offered functionality is useful for their classes, as not each class may, e.g. require a student forum.

The use of electronic media for courses also requires adjustments in the way how students learn, they only accept such a technology if they have real advantages. From our experience students rarely use forums; even in lectures with a large audience, e.g. the introductory courses in statistics with nearly 500 students, more than 99% of the entries came from the lecturer himself, a test with anonymous accounts provided same results. Most students prefer to send their questions via email or consult the lecturer personally, but even the number of emails is considerably low. In our opinion the students are afraid of asking a 'foolish' question and rather prefer the direct interaction from face to face. The use of forums even seems to cause, especially when anonymous accounts or webmailer account are enabled, a tendency to use inappropriate language. Several forums had to be closed at ISE due to insulting postings from the students.

3.4 Conclusion

A few years ago, e-learning in statistics was a *must have* for the modern education of students. Experience has shown that the goals, which are connected with e-learning elements, are different for students, lecturers and developers. The developers were interested in the technically most sophisticated solutions, the lecturers in the proliferation of their materials and the students in an easy way to achieve enough knowledge to pass the exam.

Nowadays we see e-learning as valuable support tool to aid the learning process, but the problems which may arise in the work with e-learning tools are still manyfold. Besides technical problems such as the limitation to specific web browsers there are pedagogical issues and side effects as well.

Therefore our conclusion is that:

1. e-learning cannot replace the interaction of student, teacher and blackboard
2. e-learning tools can only be successful if they satisfy the need of all participants of the system

The requirements for excellent electronic media in education are manyfold: Robust and reliable technology, high-quality contents and the willingness to adjust the own behavior from both, the students and the lecturers.

4 On the Utility of E-Learning in Statistics (ISR 2007)

4.1 Introduction

Nolan and Lang (2007) and Darius et al. (2007) both offer a technical solution to improve the understanding of statistics by students: Nolan and Lang (2007) propose a document system whose aim is to allow students follow the decision making process of a statistical analyst; Darius et al. (2007) present a set of Java applets to assist the students' understanding of experimental design.

In past decades a plethora of interactive tools has been developed with the aim to help students to receive a better understanding of statistics and to develop a 'statistical thinking'. Some prominent commercial examples are Fathom (Keypress, 2007) or Visual Statistics (McGraw-Hill, 2001). Common internet search engines deliver more than one million hits for the search term ßtatistics applet".
However this development rises the following questions:

- Why do (economics) students have those severe problems with introductory statistics that additional tools are required?
- Which impact do e-learning tools have on our teaching?
- What can we expect to happen in the future?

4.1.1 Student problems with statistics

The reason for the students' difficulties in statistics are complex and can be assigned to the students themselves and their lecturers.

For more than ten years the students in our department evaluate all given courses and lecturers. Figure 4.1 shows the difference between the average answer on the questions "General impression fo course/Course expectation fulfilled"for the introductory course in statistics and all courses in the department from winter term 1999/2000 until summer term 2006. The graph indicates that our main problem are the courses in introductory statistics and even a good teacher (marked with small black points) can not fulfill the students expectations. Nolan and Lang (2007) give a detailed example for email spam classification, but we have to ask: Do they tackle the right audience?

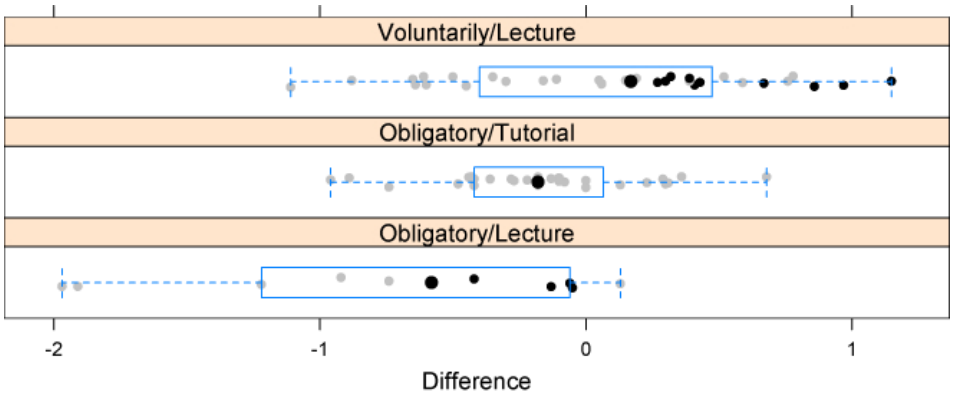


Abbildung 4.1: Difference of mean answers for statistics courses and mean of all courses to the question „General impression of the course/course expectation fulfilled“ (1=fully to 5=not at all) from winter term 1999/2000 until summer term 2006 by course attributes (obligatory/voluntarily visit and lecture/tutorial). The small black dots represent one particular teacher.

The overall average of the statistics course is around -0.12, which means that the courses in statistics are slightly lower evaluated by students than the average. The verbal comments from the students name three problems in the compulsory basic courses as well as in the voluntary advanced courses in statistics:

- too fast, too much content in the curriculum
- too formal, too much mathematics
- not enough examples

These comments led us to the development of several learning tools, with whose help we tried to overcome the problems. By distributing course materials in a convenient (electronic) form we give our students a chance to concentrate on the lecture rather than copying the contents of the slides. By providing a variety of examples that would be too time-consuming or complex for the lecture we try to support the students' understanding.

Unsolved until now and maybe forever is the issue of formality in mathematics and statistics classes. Although in our opinion the level of formal abstraction is already low especially in the introductory courses the level desired by the students is lower.

4.1.2 Online Learning Tools

In this section we will briefly introduce a few of the technical solutions we developed or in whose development we have been involved in:

- MM*Stat
- Electronic books
- e-stat
- Q&A

The first project, MM*Stat (Müller et al., 2000), aimed to supplement the traditional lectures by creating a framework to repeat and practise the contents of the lecture. Using a HTML-based filing card structure with 'lecture' filing cards providing the definitions, formulas and graphics and various example cards the student could stepwise follow the lecture. Interactivity was achieved using JavaScript-based multiple choice questionnaires and Java-based interactive applets using the XploRe Quantlet Server technology (Borak et al., 2005).

The XploRe Quantlet Server technology was also used for the electronic versions of several books published in the institute. Using links in the online version and name tags in the print version the reader could access HTML pages which allowed to run the example from the book interactively. Examples for these electronic books can be found at `http://www.xplore-stat.de`.

The experience from MM*Stat later influenced also the e-stat project, a joint venture of different German universities to create a unified platform for learning statistics.

The system offered different levels of abstraction and various views and scenarios Cramer et al. (2002):

Method-based learning of methods along a predefined way of modules with terms, examples and exercises

Problem-based based on a simplified description of the problem, a consulting component then proposes a suitable solution

View-based problem description from specific areas serve as examples to deal with the underlying terms

Especially created for the preparation of exams was the Q&A system (Klinke, 2004). Q&A allowed to check for common student mistakes by offering various pre-calculated solutions for each exercise. Depending on the student's answer the teacher could determine the source of error.

Besides these we use Microsoft Excel in our teaching as it allows the easy use of interactive slides and buttons to generate visualizations of parameter changes.

4.1.3 Impact on our Teaching

Not just the availability of additional e-learning tools has changed in recent years but also our style of teaching. For most of our classes there are sets of electronic slides, so teaching often means teaching from a PDF presentation instead of writing on the blackboard. This solution allows also the students to have the course materials in advance, however there are certain disadvantages. The lecturer cannot simply leave the media and e.g. quickly add a drawing; in many lecture halls at Humboldt-Universität a parallel use of beamers and overhead projectors is difficult since both project on the same area at the wall.

If we look at our own statistics courses, which are the tools that we use? In introductory statistics the students received the MM*Stat CD-Rom and although the students' comments were very positive an evaluation (Brandes, 2004) showed no significant improvement of the grade. Our conclusion from the result is that the CD was mainly a replacement for textbooks. Inside our tutorial classes MM*Stat was used only rarely for several reasons: at that time only few rooms at Humboldt-Universität had been equipped with beamers, the MM*Stat examples did not fit to the exercises handled in the tutorial and finally, since most lecturers preferred to teach via blackboard and overhead projector.

At the moment we use in the introductory statistics Excel sheets and we have several reasons for not using more advanced packages other than that:

Didactic concept: Often a didactic concept has never been discussed or questions concerning technical implementations received more weight in the development process. We – as many developers of e-learning tools – have not had an education in didactics.

Administrative overhead: To set up especially bigger software packages may require a lot of administrative work (security, access control) which has nothing to do with the original intention of just teaching statistics.

Specialization E-learning tools are often developed by a single person or in small teams for a special context only. Since the documentation often lacks a certain level it may be easier to implement something new.

Availability Excel can be judged as one of the packages that each of our students has access to. So in oppsite to tools which require a complex installation or use we can assume that each example we provide may be used by the students. However, we must be aware that most of our students are just "button-pushers"which means that although they know how to run a given software or how to use a Excel sheet they are not able to modify the programs or to modify a Excel sheet.

Invested effort: If we have to fear that a tool we develop for teaching will not work in a few years due to changes in the underlying operating system or programming language, why should should we invest considerable efforts. The contents of our introductory statistics courses will be most likely be the same in 2017. as in 2007.

Evaluation: We do not know any resonable study in the area of statistics where it is proven that the use of interactive tools improves the understanding by students. Most projects report for example number of accesses on project web pages etc., but no hard proof in terms of increased grades of students in a experiment is given.

All points mentioned above boil down to one question: Is the possible positive effect an e-learning tool may have on the students' understanding worth the effort he or she has to invest?

4.1.4 Future Developments and Conclusion

As statisticians we know it is difficult to predict the future. But some well known facts can give us hints to judge future developments: The PISA study for pupils has shown deficits in mathematics, especially in Germany and Austria. These results must have some effects on the next generations of students. At the Johannes Kepler University Linz in Austria some of the PISA exercises originally intended for 15-year olds were given to students in statistics courses to check their basic knowledge, e.g. in questions like "How much is 30% of 70% percent?". As Duller (2004, 2007a,b) reported the results were unsatisfying and had become worse over time. We have the same impression of our students in Berlin.

On the other hand we see a development that e-Learning tools do not target the basic statistics courses but rather more advanced course. It may of course be more challenging to implement e-learning tools for interesting subjects and classes than for introductory statistics but a good understanding of the fundamental concepts may be more helpful than a tool which visualizes just one particular highlevel-problem.

Our expectation is that the development our e-Learning tools will go on as in the past. It will replace some traditional teching methods, but a real substantial improvement can not be achieved.

5 Multi-Media and Webtools in e-Learning Statistics

The increasing availability of broadband internet also has implications on the way we can learn and teach. Technical developments allow to broadcast video in high quality, new software developments create environments to learn together in a collaborative way. In this paper we describe various web-based tools used in the teaching of statistics at Humboldt-Universität zu Berlin and show how video materials created by teachers and students can be used in the curriculum. Furthermore we introduce the Quantnet platform, a framework to manage and distribute scientific source code.

5.1 Introduction

The term 'e-Learning' does not describe a single learning method or tool it rather describes a whole family of learning practises and techniques. Following a definition by Kerres (2005), we can define *e-Learning* as all forms of learning where digital media are used for the presentation and distribution of teaching materials and/or the support of face-to-face communication. e-Learning applications are usually divided into:

- web- and computerbased training applications
- authoring applications
- simulations
- videoconferencing/teleteaching systems
- learning content management systems
- content-catalogues
- digital learning games

which can be used in various settings such as learning communities, virtual classrooms or blended learning. Since the boundaries often overlap, a clear separation between them is not possible.
In the following section we will show how the wikis can be used to work collaboratively in e-Learning projects. In Section 2 we will describe the use of the learning content management software Moodle to evaluate students' knowledge using multiple choice exercises. In Section 3 we will describe our experiences with the online publication of digital video- and audio materials before introducing the Quantnet platform, a web-based system to manage scientific source codes and data, in the last section.

5.2 Wikis

In terms of available features web-based content can clearly outperform traditional media such as books. Different parts easily be linked, dynamic content such as videos and audios can be embedded. However the manual creation of HTML pages can be tedious since texts need to be converted and linked with each other, files need to be managed. This last argument especially hold for projects with several authors.

With the goal to allow our students and colleagues to collaboratively work on web-based learning content, we have introduced wikis in our department. A wiki (Hawaiian for 'quick') is a collection of webpages that may not only be read by its viewers but also modified. Wikis are suitable especially for teams, when several people work collaboratively on a project, or in environments with fast-changing contents. Wiki software uses a, in comparison to HTML simpler syntax, which is converted to HTML by the wiki engine. See Listing 5.1 for an example, how wiki syntax can be used to mark different section and list levels.

```
1 == section ==
2 === subsection ===
3 ==== subsubsection ====
4 * list item level 1
5 ** list item level 2
6 # numbered list level 1
7 ## numbered list level 2
```

Listing 5.1: Basic wiki syntax (MediaWiki)

We use instances of the MediaWiki software for various projects in e-learning. While there are other Wiki packages available (even the Moodle course management software offers a wiki module) we decided to use MediaWiki, as this software has proven its stability by hosting the Wikipedia encyclopedia, uses LaTeX syntax to display mathematical formulas and can be extended via plugins. Furthermore the use of MediaWiki allows us to conveniently integrate our content into the Wikipedia.

The first instance, TeachWiki, is used to publish seminar and Bachelor/Master theses online. After receiving an account, the students can publish their texts online. To join different topics in statistics, the students can create link, either to the works of their fellow students or to external sources.

The second instance of MediaWiki, StatWiki, provides an online dictionary of statistical terms. The entries were based on a printed dictionary by Rönz and Strohe (1996) and the glossary of MM*Stat (see below), which have been extended collaboratively by the authors of StatWiki.

The goal of StatWiki was to create a online platform, where students can look up statistical terms using a single, reliable source with definitions written in their lingo and to which their lecturers can simply refer to. For many of the covered terms Wikipedia may provide a more extensive definition but the aim was to give brief definitions, in addition we can guarantee that an article read today is available in the same form tomorrow.

The last wiki, the MM*Stat wiki, was used to bring the contents of the MM*Stat learning environment on a modern platform. MM*Stat (Müller et al., 2000), started in 1998, was our first attempt to create digital contents for teaching statistics in undergraduate courses. The main goal was to improve the statistical understanding of economics and business administration students by using interactive features that a traditional book cannot provide such as non-linear structure by using hyperlinks, multiple choice questions and interactive examples using an embedded client-server software.

Abbildung 5.1: German edition of MM*Stat in MediaWiki

As the original MM*Stat platform was developed for 'archaic' browsers (Netscape 4 and Internet Explorer 5), modern HTML browsers showed errors in accessing some features. To be able to provide our students with the content, the HTML were merged manually into the MM*Stat wiki.

This procedure was also used to generate the Arabic version of MM*Stat, ArabMM*Stat (Ahmad et al., 2007). Since no reasonable method to convert Arabic LaTeX code to HTML could be found which paid tribute to the right-to-left writing direction and the non-latin character set, the texts and examples were entered into the wiki and linked with each other manually.

Since statistical education needs numerical examples we wanted to add interactive examples to the different wikis. Using the plugin architecture of MediaWiki we developed an extension which, analogue to the Sweave software for LaTeX(Leisch, 2002), allows to integrate **R** code into wikis.

The **R** extension by Klinke and Zlatkin-Troitschanskaia (2007) provides additional tags that will either display raw output, tables or graphics, as special `<Rform>` tag allows limited interactivity by creating HTML forms.

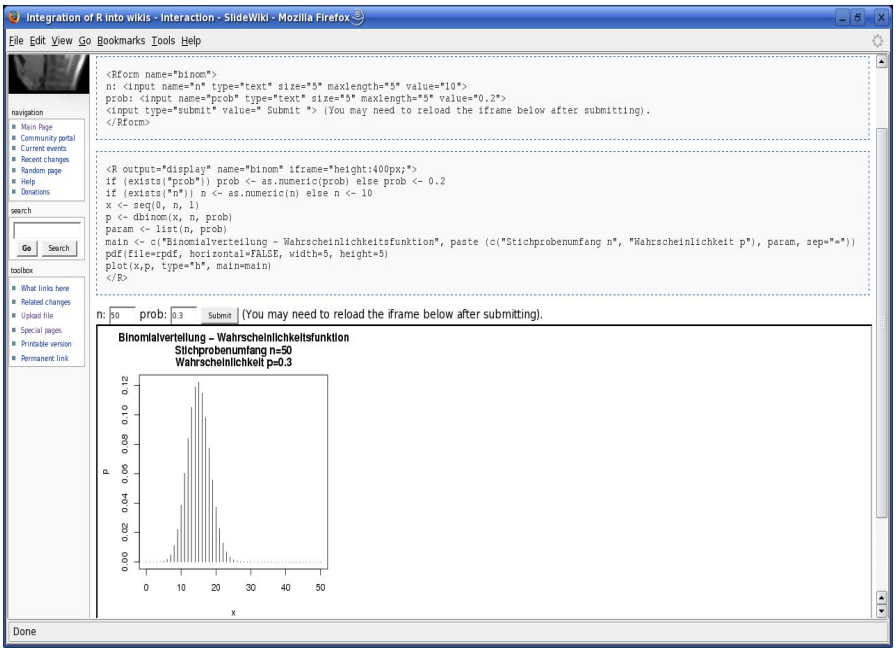
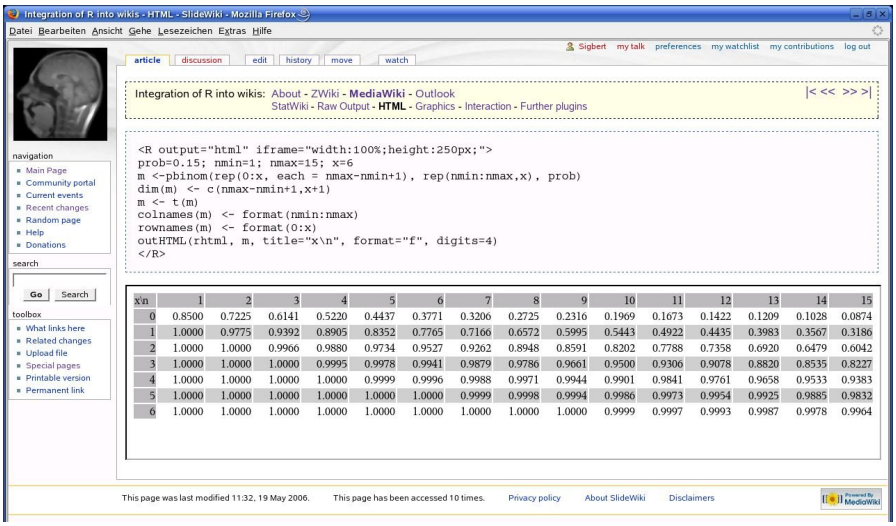


Abbildung 5.2: Interactive form for a graphics with the probability function of the Binomial distribution function computed by **R**.

With the use of wikis in teaching we made ambivalent experience. The TeachWiki has been very successful, as more than 60 students already published their thesis online whose submissions were partly used to extend the Wikipedia book on statistics (Miscellaneous, 2004). Usually our students realize the real value of the system, when they prepare their own thesis and make the connection between different topics in statistics by linking to materials from their fellow students.



x\n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.8500	0.7225	0.6141	0.5220	0.4437	0.3771	0.3206	0.2725	0.2316	0.1969	0.1673	0.1422	0.1209	0.1028	0.0874
1	1.0000	0.9775	0.9392	0.8905	0.8352	0.7765	0.7166	0.6572	0.5995	0.5443	0.4922	0.4435	0.3983	0.3567	0.3186
2	1.0000	1.0000	0.9966	0.9880	0.9734	0.9527	0.9262	0.8948	0.8591	0.8202	0.7788	0.7358	0.6920	0.6479	0.6042
3	1.0000	1.0000	1.0000	0.9995	0.9978	0.9941	0.9879	0.9786	0.9661	0.9500	0.9306	0.9078	0.8820	0.8535	0.8227
4	1.0000	1.0000	1.0000	1.0000	0.9999	0.9996	0.9988	0.9971	0.9944	0.9901	0.9841	0.9761	0.9658	0.9533	0.9383
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9998	0.9994	0.9986	0.9973	0.9954	0.9925	0.9885	0.9832
6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9997	0.9993	0.9987	0.9978	0.9964

Abbildung 5.3: HTML table of the Binomial distribution function computed by **R**.

```
# header
rfiles<-"/srv/.../Rfiles"
rpdf <- "/srv/.../Rfiles/76...c7.pdf"
n<-"50"  # these two lines appear only
prob<-"0.3" # if parameters are submitted
# user program
if (exists("prob"))
  prob <- as.numeric(prob) else prob <- 0.2
if (exists("n"))
  n <- as.numeric(n) else n <- 10
x <- seq(0, n, 1)
p <- dbinom(x, n, prob)
main <- "some headline ..."
pdf(rpdf)
plot(x,p, type="h", main=main)
# footer
q()
```

Listing 5.2: The **R** program which generates Figure 5.2 for the values $n = 50$ and $p = 0.3$.

The usage statistics of StatWiki and the MM*Stat wiki show that both systems are not much used by the students but further integration into the curricula will bring higher user numbers and more information on how both wikis can be improved in terms of content and acessibility.

5.3 Questionnaires with Moodle

Managing a class can consume significant amounts of time. Course materials need to be published, emails sent and answered, assignments prepared and evaluated.

Course management or learning content management software can free the lecturer from most this work by providing a platform to manage students, topics and materials under one roof. In this section we focus on the Moodle course management system and especially on its functions to generate online tests. While there are quiz modules even for MediaWiki their functionality is usually lower than the functionality provided by Moodle.

Moodle (`http://www.moodle.org`) is a free learning (content) management system, which supports the collaborative learning process by providing a course framework for digital content of all kind.

With Moodle lecturers can easily manage learning materials, distribute exercises and communicate with all their students. Moodle furthermore encourages the communication between teachers and students by its built-in forums. When a student asks a question, it may not only be answered by the lecturer but also by fellow students. In our experience the students, after this features had be used only rarely in past semesters, have realized the advantage of this communication and started to use the forum very frequently, to ask for help on specific exercises or to form learning groups. To evaluate the students' knowledge we have used the Moodle test module, which allows to generate online exams with questions composed of the following answer types:

Multiple Choice where the student has to choose from multiple answers.

Short Answer The student types a word or phrase, several answers may be correct.

Numerical Like the short answers but with an accepted error tolerance. This type of answer is useful for questions where rounding may play a role.

True/False The student can select either 'True' or 'False'.

Matching A list of questions is provided with a list of answers. The student must match the correct answers with each question.

Embedded Answers A passage of text that contains various answers embedded within, including multiple choice, short and numerical answers.

Calculated This question type allows individual numerical questions by using wildcards. These wildcards are then substituted when the questionnaire is taken.

The advantages of this solution are obvious: The module is perfectly integrated into Moodle, which makes it easy for lecturers as well as for students to use the tests. There is no need to spend additional time on the programming of such a framework, time which can be spent on the content of the tests instead. Moodle even delivers statistical information on how many students completed each question and how they performed thus giving important information on the students' understanding of the topic.

We have used Moodle's test module to integrate more than 300 multiple choice questions from the original MM*Stat (Müller et al., 2000), covering various topics from undergraduate statistics, see Figure 5.4 for a screenshot. To evaluate the students' answers one can use the built-in statistics, see Figure 5.5, or export all answers to CSV-files that can be evaluated with any statistical engine.

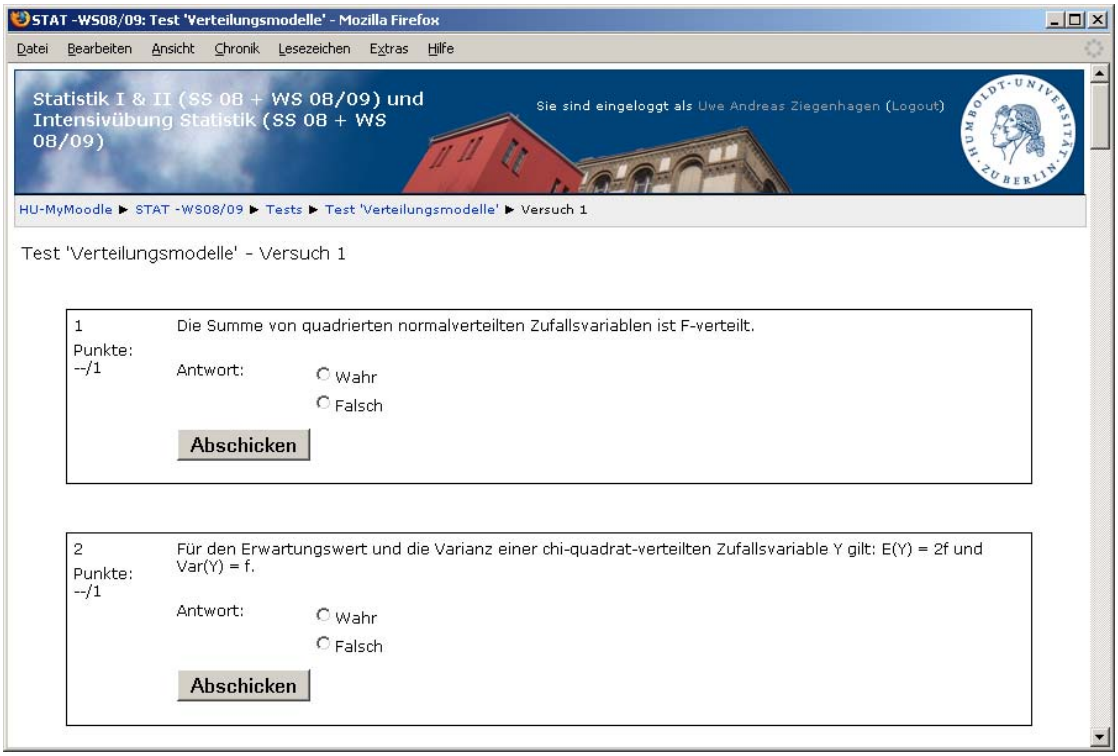


Abbildung 5.4: Screenshot of Moodle test module

Since all questions required just an answer from the students on whether a given statement was right or wrong we just used the True/False category of questions, at a later stage we plan to update the existing questions and create new ones. The amount of manual work needed to integrate these questions could be kept at low level as the old questions could be extracted from the HTML code of MM*Stat and converted into the Moodle-specific XML import format.

Before the exam a large proportion of our students did the tests we provided, based on personal feedback and the evaluation of the test results the students acknowledged this additional tool check their knowledge.

ID der Frage	Fragestellung	Antwort	Punkte	Antworten gezählt	Antworten %	% korrekt	Standard abweichung	DI	DC
(24700)	Entscheide ob richtig oder falsch. : Die Variable 'Intelligenz' ist ein metrisch/absolutskaliertes Merkmal.	Wahr	(0.00)	0/34	(0%)	15%	0.359	-0.09	0.61
		Falsch	**(1.00)**	5/34	(15%)				
(24696)	Entscheide ob richtig oder falsch. : Die Variable 'Telefonnummer' ist ein metrisch/verhältnisskaliertes Merkmal.	Wahr	(0.00)	0/34	(0%)	12%	0.327	-0.06	0.68
		Falsch	**(1.00)**	4/34	(12%)				
(24697)	Entscheide ob richtig oder falsch. : Die Variable 'Rechtsform einer Unternehmung' ist ein metrisch/absolutskaliertes Merkmal.	Wahr	(0.00)	0/34	(0%)	12%	0.327	-0.06	0.68
		Falsch	**(1.00)**	4/34	(12%)				
(24703)	Entscheide ob richtig oder falsch. : Die Variable 'Einwohner von Hamburg' ist eine Bewegungsmasse.	Wahr	(0.00)	0/34	(0%)	12%	0.327	-0.06	0.59
		Falsch	**(1.00)**	4/34	(12%)				
(24706)	Entscheide ob richtig oder falsch. : Die Variable 'Freunde der Studentin Maria' ist eine Bewegungsmasse.	Wahr	(0.00)	3/34	(9%)	6%	0.239	-0.06	0.07
		Falsch	**(1.00)**	3/34	(9%)				
(24707)	Entscheide ob richtig oder falsch. : Die Variable 'Wahlberechtigte Bürger' ist eine Bewegungsmasse.	Wahr	(0.00)	4/34	(12%)	12%	0.327	-0.06	0.51
		Falsch	**(1.00)**	6/34	(18%)				
(24648)	Entscheide ob richtig oder falsch. : Die Variable 'Geschlecht' ist ein nominalskaliertes Merkmal.	**Wahr**	**(1.00)**	3/34	(9%)	9%	0.288	-0.03	0.73
		Falsch	(0.00)	1/34	(3%)				
(24649)	Entscheide ob richtig oder falsch. : Die Variable 'Wohnsitz' ist ein nominalskaliertes Merkmal.	**Wahr**	**(1.00)**	3/34	(9%)	9%	0.288	-0.03	0.73
		Falsch	(0.00)	0/34	(0%)				
(24655)	Entscheide ob richtig oder falsch. : Die Variable 'Windstärke' ist ein ordinalskaliertes Merkmal.	**Wahr**	**(1.00)**	1/34	(3%)	3%	0.171	-0.03	0.24
		Falsch	(0.00)	1/34	(3%)				
(24659)	Entscheide ob richtig oder falsch. : Die Variable 'Jahresumsatz' ist ein metrisch/verhältnisskaliertes Merkmal.	**Wahr**	**(1.00)**	1/34	(3%)	3%	0.171	-0.03	0.24

Abbildung 5.5: Screenshot of Moodle test evaluation

The experience with Moodle is very good, the platform provides easy access for students as well for teachers. Moodle encourages the communication between all participants, teachers and students. While in earliers semesters students and lecturers had to be convinced to use the system, the added value in terms of saved time and gained functionality has convinced all users by now. With the test module Moodle offers a sophisticated way to evaluate the knowledge and to check the understanding of the students.

5.4 The use of videos in Teaching

According to learning theory the percentage of gained knowledge is significantly higher when participants watch videos instead of reading a book or paper. Different authors suggest a percentage of 50% for listening and viewing compared with 10% for reading, 20% for listening respectively 40% for viewing.

There are numerous ways to generate video content: by taping lectures, recording plays or composing animations from screenshots. We have two approaches, the creation of videos by students and the broadcasting of conference talks via podcast, which we will describe in the following.

As an alternative to standard exams we have asked students to create short animations on selected topics in statistics. The aim is that students explain topics from a lecture to other students which they had difficulties to understand. We are working together with a student initiative which provides, beside the Teachwiki, a platform for the videos; see for example `http://www.sofatutor.de/videos` *Clusteranalyse: Theorie und Praxis mit SPSS*. About one third of our students preferred this alternative way of examination.

The videos, which were generated in the Windows Media format, have a length between ten to twenty minutes and were created by the students using the Windows Movie Maker, a software shipped with Windows XP, and Snag-It, a screen grabbing software. The screenshots made with Snag-It were imported to Movie Maker and combined to a video sequence together with spoken explanations recorded separately.

For the second approach we could use highly sophisticated technical equipment, from high-quality cameras and wireless microphones to professional video editing software on a powerful workstation. The aim was, with the creation of 'podcasts', to publish our conferences for a broader audience in- and outside Humboldt-Universität and Germany.

The term 'podcast', a concatenation of the words 'iPod' and 'broadcast', describes a series of audio and/or video files, usually in MP3 repectively MP4 format, stored on a webserver. Although the technique was known before it gained immense popularity when Apple introduced the support for podcasts in iTunes and their well-known portable video and audio players.

In addition to the media files there is a so-called RSS (Really Simple Syndication) feed, an XML file which contains meta-information on each podcast episode. Listing 5.3 shows an example podcast file. Figure 5.6 shows a screenshot from the Humboldt-Princeton conference podcast series, available under `http://fedc.wiwi.hu-berlin.de/podcasts.php`, Figure 5.7 the same video on a iPod Touch mobile player. With a resolution of 480*320 pixels even small formulas or picture are clearly visible.

```
<?xml version='1.0' encoding='UTF-8'?>
<rss xmlns:itunes='http://www.itunes.com/DTDs/Podcast-1.0.dtd' version='2.0'>
  <channel>
    <title>Basic Statistics</title>
    <description>Learning statistics</description>
    <link>http://domain.xyz/feed.xml</link>
    <itunes:category text="Learning"/>
    <itunes:author>HU Berlin</itunes:author>
    <language>en-gb</language>

    <item>
      <title>Confidence Intervals</title>
      <itunes:author>Uwe Ziegenhagen</itunes:author>
      <itunes:summary>Introduction to confidence intervals</itunes:summary>
      <link>http://localhost/confence.mp3</link>
      <itunes:explicit>no</itunes:explicit>
      <itunes:keywords>Statistics, Confidence Intervals</itunes:keywords>
    </item>
  </channel>
</rss>
```

Listing 5.3: Example of a iTunes feed

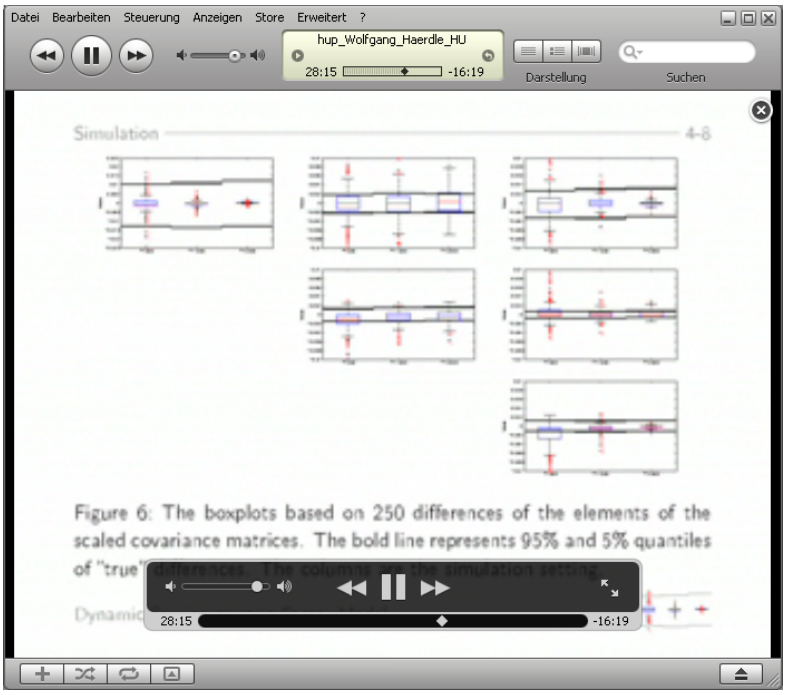


Abbildung 5.6: Podcast episode with Quicktime controls

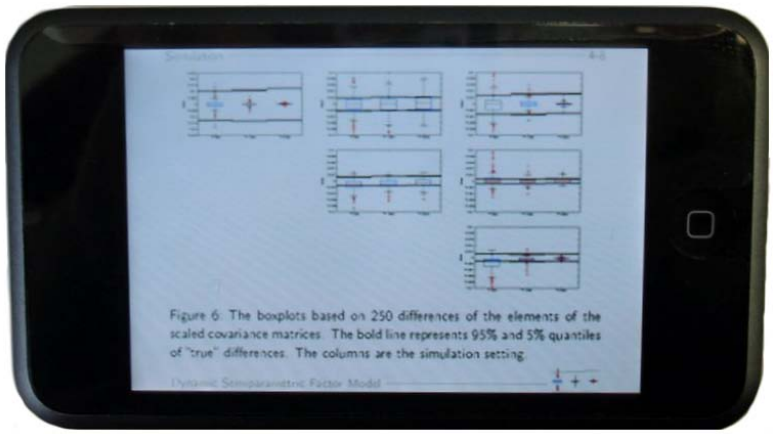


Abbildung 5.7: The podcast episode from Figure 5.6 on an iPod touch

The experiences we made with the use of videos are mixed. The recording of lectures provides a real value for the students, who are enabled to watch missed classes or can concentrate on the class as there is no need to write everything down. It allows also to reach a broader audience by broadcasting interesting events online.

However, the time and energy to create good content may be extraordinary high. Only few classrooms are equipped with the necessary video equipment by default and usually there usually is no trained personnel to record and produce video content. So

it mostly depends on the lecturer's energy and expertise to record and broadcast his or her lectures. Concerning the videos created by our students this argument holds as well. Compared with writing a seminar thesis a short video may require much more time and technical expertise.

5.5 Quantnet

The founding father of Humboldt-Universität, Wilhelm von Humboldt, argued for a *Unity of Research and Teaching*. While this term clearly had a different meaning in his days, we can use it today as ambition to use currently published research for teaching our students.

A common problem in publishing research results is to provide the methods, codes and data for the analyses which allow other researchers to reproduce the analysis and students to learn from the provide materials. Based on earlier works, among them the XploRe help system and e-book system described in Klinke and Witzel (2002) and the MD*Base data library `http://www.quantlet.org/mdbase`, we want to establish *Quantnet*, a framework for the storage and documentation of scientific code, which collects works from researchers as well as from students under one roof.

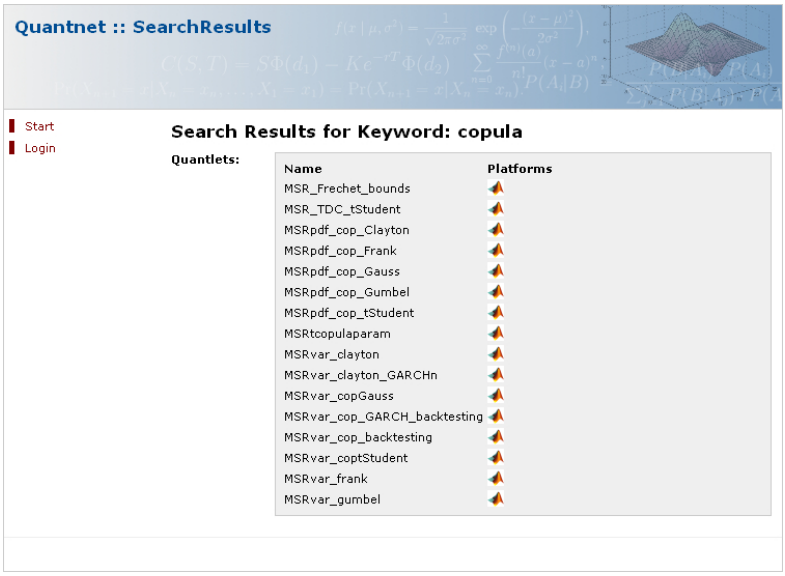


Abbildung 5.8: Quantnet screenshot listing all code with the 'copula' keyword

The Quantnet project, see Figure 5.8 for a screenshot, takes the experience made with MD*Base and XploRe and generalizes their ideas by allowing methods and algorithms implemented in various scientific engines such as Matlab, **R** or Gauss. The project homepage is available under `http://fedc.wiwi.hu-berlin.de/quantnet`.

In the background the Quantnet system uses a relational database to store all content. This has the advantage that with the use of SQL queries one may easily select specific

codes e.g. all codes by Author 'Doe' or all materials which belong to a certain book or thesis. The submission to Quantnet is simple, the user may upload single files or zip archives following a certain structure with information on the author, topic and usage of the material. The sources are then automatically evaluated by the Quantnet system and, after validation through the author himself, stored in the database. Figure 5.9 shows a screenshot of the page for a single example.

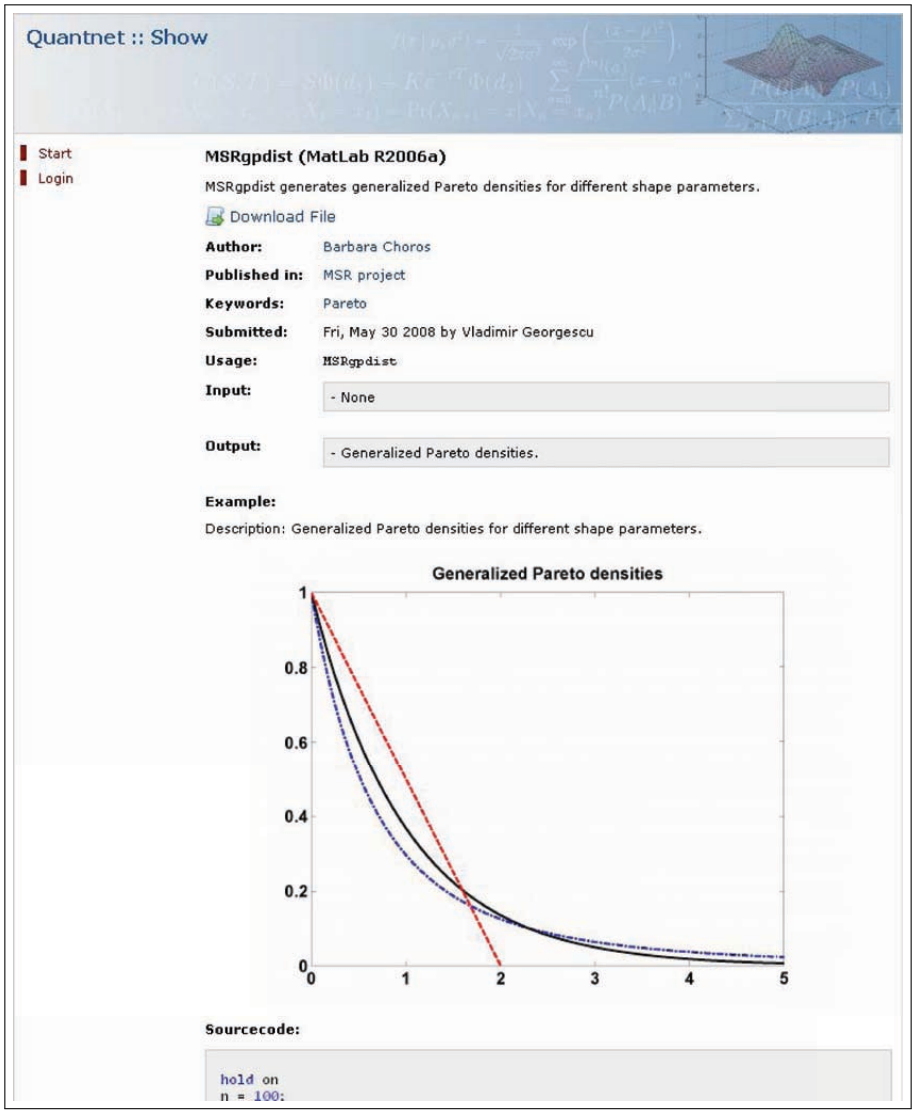


Abbildung 5.9: Quantnet screenshot of a single code file

At the moment the system is able to handle Matlab and **R**, the support for other scientific languages is on the agenda. A planned extension of Quantnet includes the integration of convenient search functions and mechanisms to group 'similar' codes based on contextual analysis.

As the system has not been online for long, the experience with the platform is still limited. Based on personal experience and the feedback from some users there is clearly a demand for platforms like Quantnet. In the near future we will expand the system in terms of functionality and hosted codes, further and more detailed evaluation will then show us how Quantnet can be integrated best in our curricula.

5.6 Conclusion

In this article we described various web-based methods to provide e-Learning elements for modern curricula. Although it is not clear which of the used techniques may provide good value for teaching in the long run some trends become apparent.

Wikis allow an easy way of working collaboratively with several authors, nice way of putting things online also for lecturers and students with little expertise in computers.

Moodle, or learning content management systems in general, provide a real added gain by saving valuable time and offering the students a standardized platform to discuss and download materials.

Videos and animations can help students with the understanding of difficults topics or allow interested persons from all over the world to participate in interesting events, however the required work are easily underestimated.

Quantnet, or similar systems, can be very helpful by collecting data and code under one roof. The future will show if researchers and students accept this system and how it may be improved. In the following table we have tried to summarize our experience in terms of in- and output for the students, the benefits and the costs in terms of labour and time.

	Wikis	Moodle	Videos	Quantnet
allows student input	+++	+	++	++
value for students	+++	+++	+++	++
benefits for lecturers	++	+++	+/-	++
labour costs/req. time	+	+++	---	+++

6 Yxilon - Designing The Next Generation, Vertically Integrable Statistical Software Environment (Interface 2004)

Before we discuss the future of statistical software it is worth to look at past and present tools for a statistican's daily work.

Back in the 60s of the last century, where several commercial software packages originally date from, data analysis was an uphill struggle. Computers were scarce, expensive and, compared with the latest generation you can buy in each supermarket today, extremely slow. Furthermore they had to be operated by specially trained personnel, often computing time even had to be reserved in advance.

When we look at the market for statistics software today, we find a large variety of free or commercial packages as S-Plus, SPSS, Minitab and R, being able to handle large data in a split second. But really astonishing is, the product used for most analysis tasks worldwide is Microsoft Excel, although there are certain doubts about its numerial precision (McCullough and Wilson, 1999).

Why is Excel thus attractive for so many users? Following Chambers (2000) we can split the job of analysing data into three different subtasks: Organisation, Analysis and Presentation, in this sequence most analysis jobs might are run today. Table 6.1 tries to point out the pros and cons in these three subtasks. We see, Excel may be a good choice for small or medium tasks, but may not be sufficient for a deeper analysis of large datasets. Here "dedicatedßtatistics packages find their right to exist. The ability to compete against a standard spreadsheet package may be taken as one requirement but what other requirements does a statistics package have?

Tabelle 6.1: Performance of Excel in Organisation, Analysis and Presentation

	pro	contra
Organisation	large variety of import & exportfilters, filter and database functions	tables limited to 65.536 rows and 256 columns
Analysis	numerous functions for statistical analysis, e.g. for ANOVA, Fourier Analysis, Regression and Sampling	numerical inaccuracies
Presentation	graphics connected with data dynamically	no statistical displays as boxplots, histograms, etc.

6.1 Requirements for Statistical Environments

Chambers and Lang (1999) give a list of features, that are desirable for statistical tools:

1. Usable from multiple front-ends
2. Support for development of GUIs for different audiences
3. Extensibilty on language/interpreter level and native core level
4. Internet abilities to read and write data to networks
5. Database support to allow dynamic analysis
6. Interactive graphics and connection to other graphical controls
7. Support for multi-processor machines
8. Extensibility by inclusion of existing code
9. Optimization for Performance

Based on our experience we can add several items:

Set of methods: The included set of methods is for sure one of the most crucial points when selecting a statistical engine. Although there is a common subset of methods, most commercial software programs show huge differences in the included method sets, most producers provide (expensive) add-ons for special analysis tasks.

Multiple language support: English is the lingua franca of science but for non-native users the *usability* of the software increases significantly when graphical user interface, hints and especially error messages are given in their own tongue.

Valuable user ressources: The available user ressources as manuals in printed and electronic form, tutorials and on-line help are the access key to the software. While experienced users often need just an index of the available functions, novices and students require more assistance in the form of tutorials and sufficient manuals. These should also provide substantial help on the theoretical background of the available methods, since, as Tukey (1965) stated: "Most uses of the classical tools of statistics have been, are, and will be, made by those who know not what they do.".

6.2 Why do we need vertical integration?

We define vertical integration as the smooth integration of statistical computing frameworks into completely different environments as web browsers, electronic or printed books and standard application software. Table 6.1 depicts three examples, how this vertical integration may look like in practise: In scenario 1 Microsoft Excel reads data from a file or database and hands them over to an embedded computing module, for the graphical presentation of the results internal Excel routines are used. Scenario 2 shows how modern web standards as XML and SOAP may be used in a vertically integrated environment. Imagine a financial service provider as Thomson Datastream or Bloomberg providing daily option data as *webservice* (`http://www.w3.org/TR/wsdl`), embedding the information in so called 'XML envelopes'. A computing engine retrieves these data and calculates a trading signal for the analyst. The last scenario uses special commands in LaTeX-source to generate output formats as HTML and PDF with embedded links to interactive examples that are run inside a webbrowser or Java applet.

Abbildung 6.1: Three scenarios of vertically integrated software

	Scenario 1	Scenario 2	Scenario 3
Organisation	Excel	SOAP/XML	LaTeX
Analysis	computing engine	computing engine	Applet
Presentation	Excel	trading signal	webbrowser

With common statistical software packages one may find it difficult to implement these scenarios, since they are mostly monolithic, offering organization, analysis under one roof. Often these packages themselves or significant parts of them have been written decades ago. As mentioned in (Theus, 1998), the basic graphics routines in *S* celebrated their 30th birthday in 1998. Reimplementation would sometimes be necessary but seems impossible for one of the following reasons:

- The original programmers are not available anymore and left little or no information on the implementation details because no documentation standards had been defined.
- A growing codebase and pool of methods created interdependencies between e.g. computing engine and graphical user interface, so changing one part of the software may affect other parts as well.
- The original design approach did not allow certain extensions, their more rough than ready implementation causes problems with performance
- Retaining compatibility to previous releases may force the developers to keep outdated code

Vertically integrable software can help us to solve these problems. When all vital parts of the software framework are designed from scratch to work together smoothly via

standardized interfaces, the modification of single parts is for sure much easier than working in a monolithic environment.

But even when the software is actually made of modules there may be problems arising. Statistical frameworks using client-server architecture largely depend on the underlying communication protocols.

Figure 6.2 shows MD*Crypt, the TCP/IP based communication architecture used by the *XploRe Quantlet Server* to exchange information with its clients. While TCP/IP has the advantage of being available for all major platforms and allowing a reliable and relatively easy way of communication, there is a drawback in speed especially when the server application runs on the same machine as the client. Using here technologies as shared memory usually brings a dramatic increase in performance.

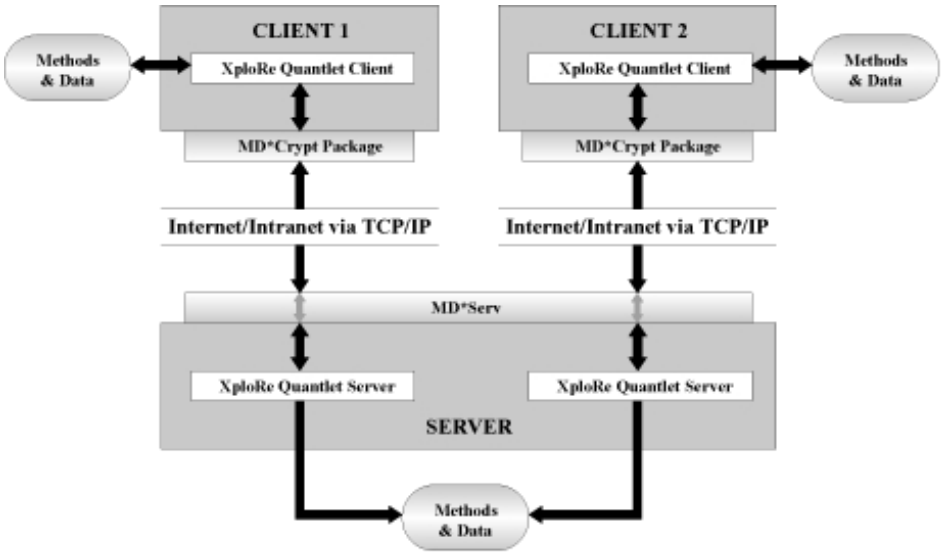


Abbildung 6.2: The MD*Crypt communication structure (Lehmann, 2003)

So the problem of communication can also be solved by vertical integration. By offering different communication protocols or interfaces, where user defined protocols can plug themselves in, gives each user to select the best appropriate trade-off between speed and user friendlyness.

Last but not least the user interfaces offer opportunities for vertically integrated software. Item no. 2 from Chambers and Lang (1999) suggests to offer several user interface instead of a single one. A monolithic software with a hard-wired cannot satisfy this requirement. So a vertical separation of computational and presentation part is needed.

The question of user interfaces becomes especially interesting, when we take usability into account. There are different definitions for the term *Usability*, a common one is given in Nielsen (2003), where usability is defined as "the measure of the quality of the user experience when interacting with something – whether a Web site, a traditional software application, or any other device the user can operate in some way or another". Nielsen (1993) divides the term üsabilityïnto five components:

Learnability How easy can first-time users accomplish basic tasks?

Efficiency How quickly can users perform tasks once they know how to use the software?

Memorability When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Errors How many (severe) errors do users make, how is the prevention of errors supported?

Satisfaction How pleasant is it to use the software?

Many software packages were originally written to satisfy the needs of either the programmer himself or a number of experienced and skilled professionals, so usability concerns were rarely taken into consideration. Since the creators of the software knew how to interact with their own piece of work, they usually did not care how non-expert users would perceive the interface offered by the software.

Research in usability and standardized test procedures for usability as done by Jakob Nielsen (`http://www.useit.com`) are relatively unusual even today, so the "Hall of Shameïn interface design (Theus, 1999) is full of examples how not to design a user interface.

For answering the question, how a good interface design might look like, the "golden rulesöf interface design (Shneiderman, 1997) may be taken as a guideline:

1. *consistency* The user expects similar reactions from the software in similar situations. When programmers work together in a team and no guidelines are given, each of the programmers implements his idea of a suitable reaction to actions from the user.

2. *shortcuts and feedback* While a beginner may need guidance in usage, the experienced users usuallys want to finish their task as quickly as possible. New users are mostly grateful for receiving feedback from the software, when certain tasks are finished but the professional user does not need this information.

3. *closed actions* Actions to be executed by the user should have well-defined start and end points.

4. *error handling* Error messages should be as short *and* informative as possible, "There was an errorïs not sufficient.

5. *loss of control* Users prefer to act, not to react when working with software. Pure reacting may be perceived as loss of control.

6. *limited short term memory* Humans are able to store just a few items in their short term memory (Miller, 1956). The user interface should prevent the user from having to memorize different settings.

6.3 The Yxilon project

6.3.1 XploRe

Since the design of Yxilon is largely influenced by XploRe, we would like to give a short overview. XploRe was developed jointly by Humboldt-Universität zu Berlin and MD*Tech, a German software company, with the aim to provide a general purpose statistical computing environment for the quantitative analysis of data. XploRe quantlets cover a wide spectrum of statistical methods as Generalized (Partial) Linear Models, nonparametric methods (kernel estimation), single index and generalized additive models, ANOVA, etc. A focus lies on financial engineering functions for option pricing, Value-at-Risk and hedging strategies.

```
proc()=SFEgamma()
; beginning of procedure SFEgamma
       s=100            ; stock price
       k=100            ; exercise price
       r=0              ; interest rate
       v=0.25           ; volatility
       tau=0.5          ; time to maturity
       q=0              ; dividend rate

    zeichen1="First variable, lower bound:"|"upper bound:"
    zeichen2="Second variable, lower bound:"|"upper bound:"
    values=50|150 ; predefined values first inputbox
    ss=readvalue(zeichen1,values)
    s1=ss[1]
    s2=ss[2]
    sw1=(s2-s1)/30

    values=0.05|1.0 ; predefined values second inputbox
    ss=readvalue(zeichen2,values)
    t1=ss[1]
    t2=ss[2]
    sw2=(t2-t1)/30
    lauf=grid(#(s1,t1),#(sw1,sw2),#(31,31))
    tau=lauf[,2]
    s=lauf[,1]

  d1=(log(s./k)+(r-q+v^2/2).*tau)/(v.*sqrt(tau))
  opv=(exp(-q.*tau).*pdfn(d1))./(s.*(v.*sqrt(tau)))
      dat= lauf~opv
      gs = grsurface(lauf~opv)
      plot3d(1,gs)
endp
; end of the procedure

library("xplore") ; load libraries
```

```
36 library("plot")
37 setsize(600,450) ; set display size
38 SFEgamma() ; call procedure
39 setgopt(plot3disp,1,1,"title", "Gamma","border",0) ; change layout of
       plot
```

Listing 6.1: XploRe code to plot the Gamma of a Call option

XploRe features a matrix-oriented programming language with C-style syntax and most of the internal functions are written in the XploRe language. Figure 6.1 depicts an example for the plotting of *Gamma* in the XploRe language, interested readers may refer to Härdle, Klinke, and Müller (2000). External procedures written in C/C++ or Fortran may be executed from within XploRe via dll function calls. An extensible HTML-based help system (Klinke and Witzel, 2002) offers detailed descriptions of all built-in functions.

XploRe has been implemented for Microsoft Windows and UNIX-based operating systems as Solaris and Linux in several versions as stand-alone, batch and Client-server version, a demo version may be downloaded from `http://www.xplore-stat.de`. A strong focus has been put on the scalability for different purposes, examples are the *XploRe Quantlet Client* (Lehmann, 2004) and *MD*ReX* (Aydınlı, Härdle, and Neuwirth, 2003), an Add-In for Microsoft Excel.

6.3.2 The XploRe Quantlet Client

The XploRe Quantlet Client is a Java-based software that, besides running as an application, also runs as applet from any Java-enabled webbrowser on any hardware platform supporting the Sun Microsystems JAVA framework.

The main goal in the development of XQC was to support teaching in several ways: It is used in projects for undergraduate students as MM*STAT (Müller, Rönz, and Ziegenhagen, 2000), books as Härdle and Simar (2003) and Härdle, Franke, and Hafner (2004a) or in a variety of electronically published books (`http://www.xplore-stat.de/ebooks/ebooks.html`).

For this purpose the XQC implements the idea, also proposed by Chambers and Lang (1999): to offer different user interfaces to different groups of users. Via configuration file the XQC can be restricted to show only a subset of its features, for details see Table 6.2. This feature was implemented since different types of users have different needs. The goal of students from undergraduate courses in basic statistics is to see changes in results when parameters, e.g. for a distribution, are made. They are mostly not interested in the implementation details while advanced students may also want to explore the sourcecode to use it in their own data analysis tasks.

Tabelle 6.2: Keys in XQC configuration file with their effect

Key	effect
ExecuteProgram	loads XploRe code ("quantlets") from a file but does not show its sourcecode to the user
OpenInEditor	opens the quantlet in the built-in editor and allows modification by the user
ShowCommandWindow	defines, whether the command window is displayed that allows direct input of XploRe commands
ShowOutputWindow	The output window for textual output can be suppressed as well, this has proven useful when only a graphics is desired as output.

The integration of interactive examples can be made either manually by changing the respective HTML-pages by hand or automatically be inserting certain commands into the LaTeX-source of a book or script. Using this MD*Book i technology by Klinke and Lehmann (2003) different output formats as Postscript, PDF, HTML and a special HTML version enriched with Javascript can be created from one LaTeX-source.

The graphical user environment of XQC resembles the Windows stand-alone version of XploRe and most graphical functions f the stand-alone solution are supported. The XQC can be used or downloaded from `http://www.xplore-stat.de`.

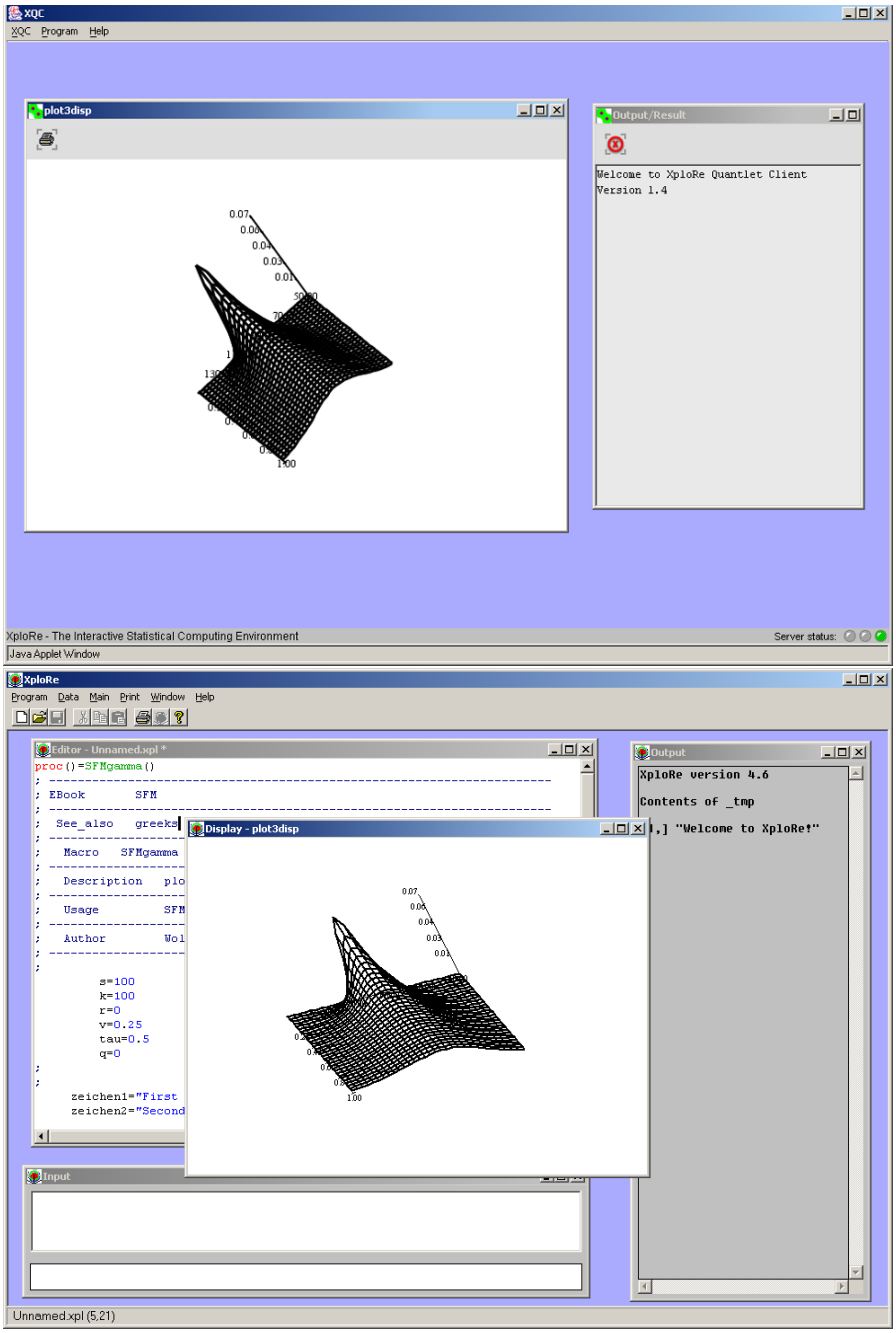


Abbildung 6.3: The output of SFEgamma.xpl (Figure 6.1) in XQC (execute) and Windows standalone version

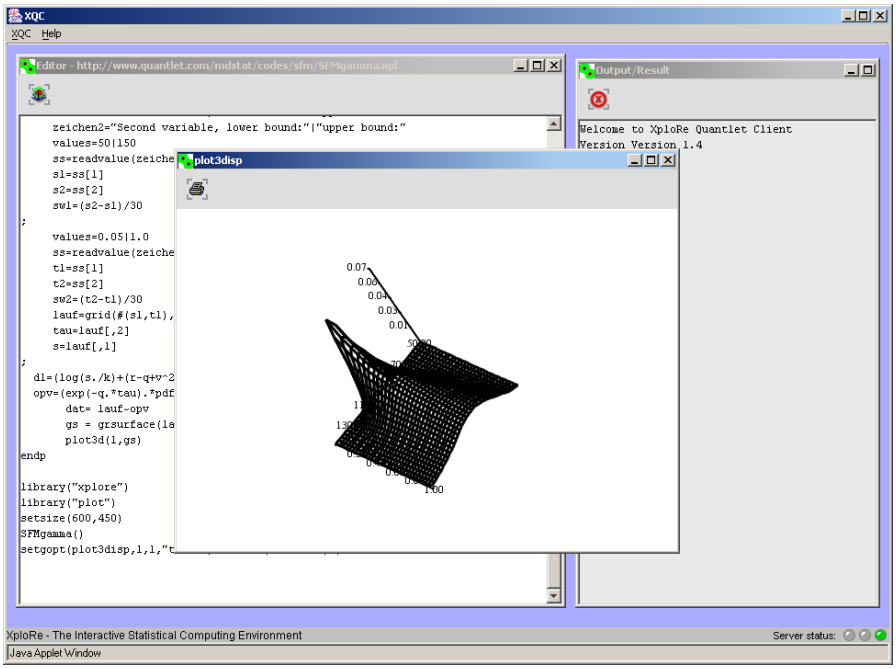


Abbildung 6.4: The output of SFEgamma.xpl in the XQC edit-version

6.3.3 MD*ReX

The MD*ReX framework (Aydınlı, Härdle, and Neuwirth, 2003) uses the *Common Object Model*, a standardized architecture developed by Microsoft. The COM technology allows objects to communicate with each other regardless of which language they are written in or on which machine they are located. Since COM is part of all available Windows versions and integral part of the Microsoft office applications, it allows a smooth integration into these application. From a technical point of view MD*ReX serves as in-process COM Server and as the XploRe Quantlet Client it uses the MD*Crypt protocol as communication layer.

Figure 6.5 shows a screenshot of MD*ReX embedded in Excel. By the additional toolbar the user can connect to local or remote XploRe Quantlet Servers and store/retrieve data. The advantage of MD*ReX is the smooth integration into Excel, only little additional knowledge is needed to work with the Excel-MD*ReX combination.

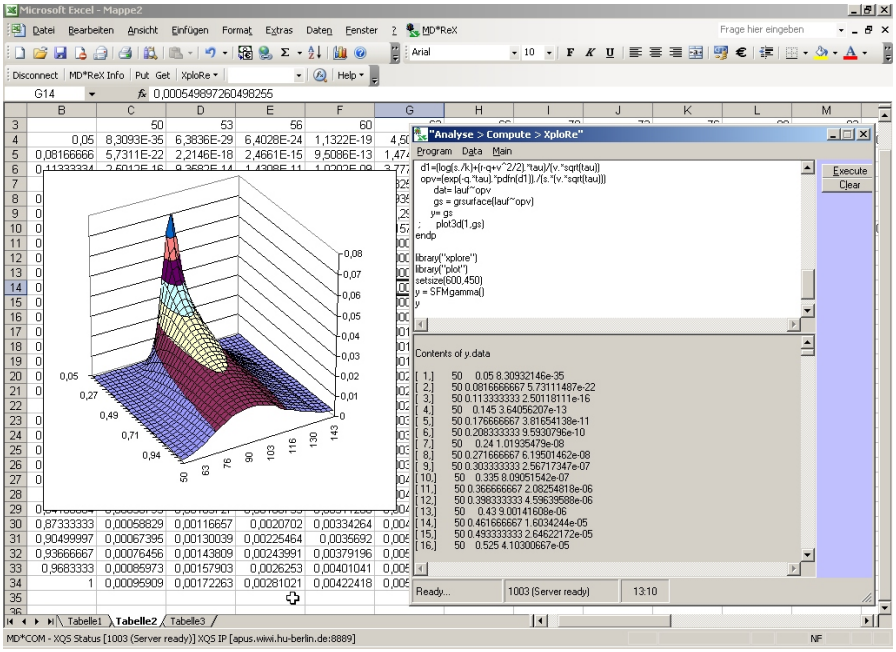


Abbildung 6.5: MD*ReX running with Excel 2003

6.3.4 Why a new software?

During the development of XploRe we faced several of the problems mentioned above. To keep and expand our expertise in statistical software we decided to form the Yxilon project. With the only premiss to be compatible with existing quantlets, we decided to change the major parts of the XploRe structure, in particular concerning:

- a strict separation of (G)UI and computing kernel, communication via light-weight protocol
- the demerger of data structures and reduction of kernel functionality
- a package and documentation system focussing on standard web techniques as ZIP and XML
- published under Free-BSD license (`http://www.quantlet.org`)

Before we examine some of these points in detail we can get an overview of the Yxilon architecture from Figure 6.6. The Yxilon core is formed of:

Object database: storing the data objects (lists, matrices, quantlets) for further usage

Parser: analysing the quantlet code, either generating C++/Java source or calling the interpreter

Interpreter: executing quantlet code directly

Database import/export filters: technically these import and export filters are clients without own user interface

Clients: the other group besdes the database plugins, including non-graphical and graphical user clients

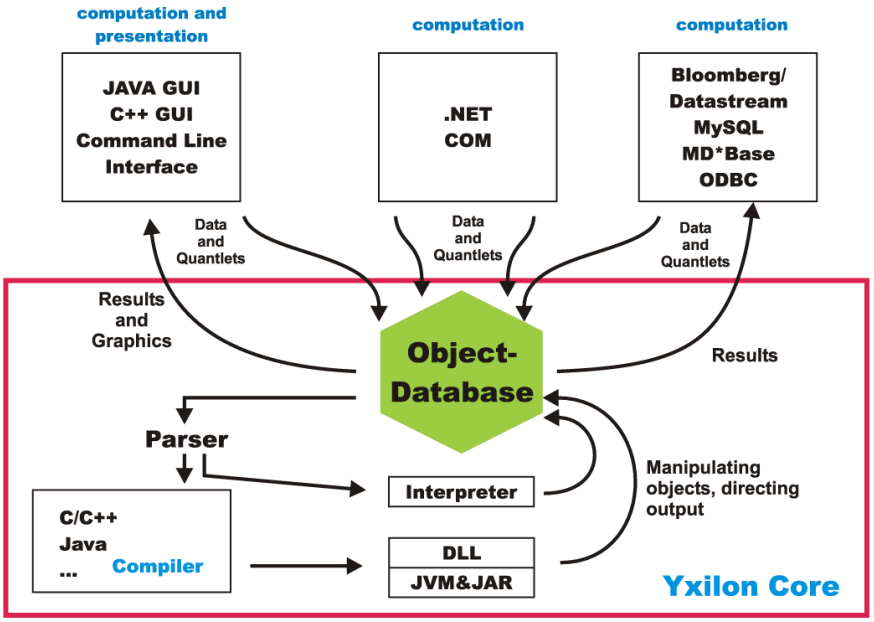


Abbildung 6.6: internal structure of Yxilon

6.3.5 Separation of Computing and Interface Components

The separation of computing engine and visualisation, that was already started with the client/server system of XploRe Quantlet Server and Client, is the basis for the communication of all clients and the computing kernel in Yxilon.

Compiled into the database component or situated directly before will be the server component dispatching the information flow from and to the different clients. As mentioned above the binary socket communication via the proprietary communication protocol has proven to be relatively slow since a unavoidable TCP/IP overhead has to be used for each data package. The conclusion is that at least for local environments where client and kernel run on one machine, faster methods as shared memory access have to be implemented. This solution has also been used by JStatCom (Krätzig, 2004), a framework for econometric routines from our institute.

For remote connections the existing MD*Crypt protocol has to be evaluated and compared with competing technologies as RPC, RMI and Corba:

RPC Remote Procedure Calls, developed by SUN Microsystems and language/processor-independent. Can only use native datatypes that have to be converted for diffe-

rent machines. Has the disadvantag of not being supported by Java natively and requiring more overhead than RMI.

RMI Remote Method Invocation, a relatively simple solution to access remote functions, but mostly limited to Java.

CORBA Common Object Request Broker Architecture, a more complex framework compared with RMI, implemented for different programming languages on several architectures

SOAP One of the newer language independent protocols, using established web standards as HTTP and XML. The client and the server transmit parameters and results via XML, this holds the disadvantage of SOAP as well: The representation in XML may even triple the amount of data that has to be transferred, on both sides the XML has to be evaluated.

6.3.6 Compilation versus Interpretation

While XploRe is a completely interpreted language, Yxilon will use in the first step of the implementation generated and compiled code. The quantlet code provided clients is parsed and a tree is set up. On the basis of this generated tree, a 'treewalker' routine generates Java respective C++ sourcecode which is stored on the harddisk. The client afterwards calls an installed JAVA or C++ compiler that creates the native code or Java bytecode.

Important is here, also concerning usability, that the compilation process is hidden from the user. There must be no difference compared with an interpreting solution as it was provided by XploRe.

In the second step an interpreter will be written, since for small programming tasks the write-compile-test-recompile cycle is inefficient.

6.3.7 Demerging of Data Structures and Reduction of Kernel Functionality

The data storage unit in XploRe was tied closely to the parsing and interpreter parts of the kernel. The final goal in Yxilon is to have the database running non-stop as service or demon with clients connecting to it. In between we implement a solution that allows the clients to create own instances of the object database to store the necessary data.

Each software needs to have specific data structures to store internal settings as paths or the size of output graphics. In principle there a two methods to access these datastructures, directly via setting flags or values as done by Gauss or indirectly via commands that change the values.

XploRe for example uses two commands to modify internal settings: `getenv` retrieves the current settings and `setenv` stores new values. In Yxilon we want to modify these commands and rather use global lists to store these values instead.

Further measures to reduce the number of commands the core needs to hold are the merger of commands and the outsourcing to DLL files. To add elements to a list

XploRe uses `append`, to add elements to a vector $\sim$ and $|$ and to concatenate strings the + operator.

The complete restructuring process includes 52% of all internal commands to be outsourced to dll-files and 24% changed from a object-oriented access to direct access structures.

6.4 Summary

Yxilon is our proposal to answer the question of how future statistical packages may look like. We will fulfill the requirements for statistical computing environments proposed (Chambers and Lang, 1999) in the following way:

Multiple front-ends: There will be two groups of frontends, user-oriented as graphical and non-graphical user interfaces as well as modules that can be vertically integrated in other standard software environments.

Different GUIs: A Java GUI in the first step, a C++ version to follow.

Extensibilty: Implicitly given since Yxilon is a full programming language.

Internet abilities: Reading and saving quantlets and data from remote locations via standard protocols.

Database support: Database support in form of DLL and JAR files.

Interactive graphics: Interactive graphics will be handled individually by each GUI.

MP Support: Thread-based models at least for the JAVA code generation

Optimization for Performance: Usage of compiled code instead of interpreation

We plan to have two stages in the development process, step one includes the completion of parser, code generator for Java and C++ and a Java-based GUI. Furthermore the interface definitions for importing and exporting data have to be made in this step. The second step includes the implementation of the database as service or demon as well as the development of a C++ based graphical user interfaces.

Details and downloads of the project are updated regularly and can be found at `http://www.quantlet.org`. A first impression on the parsing and code generation components of Yxilon can be found at `http://141.20.100.252/yxilon-j/yxilon-j.html`. We are looking for feedback from users and programmers and invite them to join us in this project.

7 Yxilon – a Modular Open-source Statistical Programming Language (ISI 2005)

Although the ASC software directory (http://www.asc.org.uk) lists far more than 100 dedicated statistical software packages for numerous purposes, hardware platforms and usergroups, the number one package used for statistical analyses is Microsoft Excel. Despite the fact that it is no dedicated statistics engine the majority of statistical amateurs and professionals feels comfortable using it also for statistical purposes. Based on our experience of nearly 15 years in developing statistical software we implement Yxilon, a new statistical engine, with the aim to be as userfriendly as a spreadsheet while providing powerful tools in diverse architectures.

7.1 XploRe

Nearly 15 years ago the first versions of XploRe were implemented. In opposite to more mouse-oriented packages as SPSS and Minitab XploRe always targeted on a language based user interaction. While this approach inheres a usually flatter learning curve in comparison with mouse-based interaction we have learned from lectures and user feedback that the users receive a deeper understanding of the underlying theories and are more actively integrated into the process of analysis.

The XploRe language is akin to C but instead of being compiled to machine code it is interpreted directly. Methods are stored in so called quantlets, written in the XploRe language, and can be bundled in libraries, the so called quantlibs.

```
1 library("plot") ; load plot library
2 x=read("pullover") ;read data
3 x=x[,2|1]     ; take columns 1 and 2
4 regx=grlinreg(x) ; compute lin. regression
5 plot(x,regx)  ; plot data and regression line
```

Listing 7.1: XploRe code to plot a linear regression for the pullover data

During the last years a multitude of new technologies also found their way into statistical science: World Wide Web, Client/Server Computing and XML were among them. To keep up with these developments, extensions as the server component *XploRe Quantlet Server/Client* Lehmann (2004) and *MD*ReX* Excel Add-In had been implemented in XploRe. Although this provided access to new features, it also increased the complexity of the program itself and the underlying source code, since the original design did

not include the transmission of data and methods via networks. At this point the XploRe programmer group decided to challenge the existing approach and to question the requirements for statistical software environments in general.

(Chambers and Lang, 1999) provide a list of requirements for statistical applications, among them: usability from multiple front-ends, internet abilities to read and write data to networks and database support to allow large-scale analysis. In Härdle et al. (2004b) we added further requirements

- modularity and extensibility
- support for multiple languages
- valuable, integrated user ressources

Since the existing approach made it significantly difficult to implement these requirements iin XploRe, we decided to reimplement the language in a completely new but compatible way, thus forming the ideas behind Yxilon.

7.2 Yxilon

While XploRe interpreted the source code we leave this approach behind in favor of compilation to native machine respective byte-code. An interpreter will be implemented as well but the main focus lies in the generation of platform-independent JAVA or C++ code, that can be compiled using the Sun Microsystems JAVA SDK respectively C++ compilers as GNU C++. The main advantages are the improved runtime behavior for computer-intensive methods and the easier inclusion of methods provided by Yxilon in different frameworks as office and publishing environments.

The following picture depicts the structure of Yxilon: All objects (methods, data, help files) are stored in a persistent object database. This object database has two main jobs: On the one hand it supplies the user oriented clients (graphical and non-graphical) and sores and retrieves data from and to a variety of sources; on the other hand it controls the internal information flow: Stored source can either be interpreted directly or transmitted to the parsing unit, that generates the C++ and JAVA code. We have chosen a modular environment rather than a monolithic application to allow easier exchange of single modules and to provide Yxilon computing power also in the form of e.g. DLL and .Net components.

7.3 Yxilon Architecture

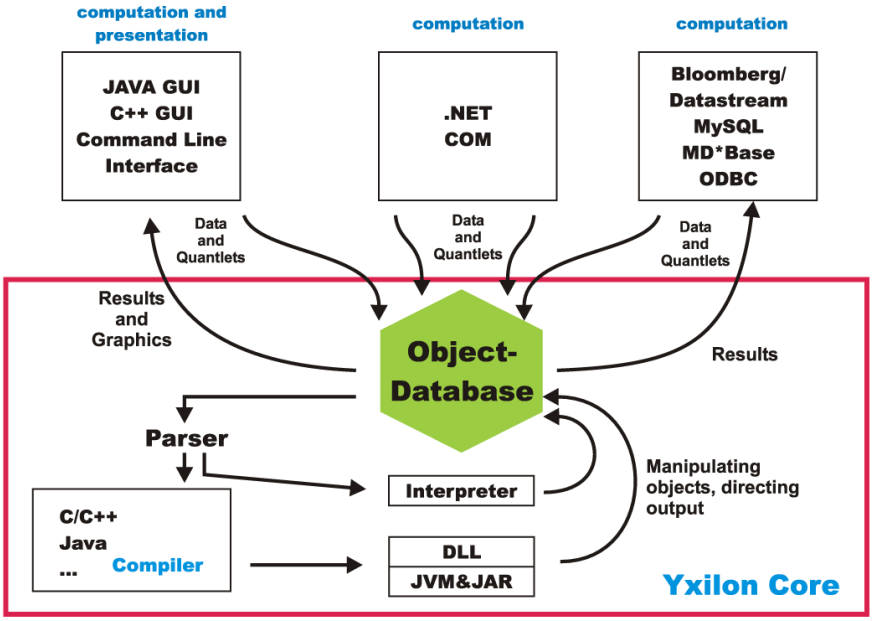


Abbildung 7.1: Yxilon Architecture

In the current stage we are working on two main tasks. One task is to specify the necessary architecture and reference implementation of the parser units, the other task is to implement a graphical user interface.

7.4 Yxilon Java GUI

The aim of the current graphical user interface is to satisfy two major needs:

- integrating the parsing and interpretation modules under one roof
- providing one experimental platform to analyze the different needs of different types of users
- implementing and testing efficient ways of interaction and communication

In the analysis of user behavior we mainly focus on the central items of *usability* Nielsen (1993): Learnability, Efficiency, Memorability, Errors and Satisfaction. (Shneiderman, 1997) furthermore defines 'Golden Rules of Interface Design', including questions concerning common rules (consistency, error handling and feedback) in the communication between user and software as well as psychological aspects (loss of control in interaction and limited human short term memory capacity).

Working especially with language based statistical software requires a significant **commitment** from the user, especially in the initial learning phase. With log4j from the

Apache Foundation (`http://www.apache.org/log4j`) we have a powerful tool to log each interaction between user and software, together with questionnaires and interviews we hope to analyze and improve the user/software interface.

The layout and design of the Yxilon gui is inspired by the current XploRe 4.7 standalone version. Nevertheless the requirements and design principles mention earlier are considered as well. All window and menu captions are provided via initialization file. The Unicode support in Java make the adaption to different languages easy. Furthermore the initialization file holds the relevant settings for the Java and C++ compiler. Our aim is to hide as much as possible technical details from the user, our premise is to confront the user with the XploRe/Yxilon language, not with implementation details of C++ and Java. This approach inheres possible error sources, if the Yxilon code is wrong either in semantic or logical dimensions the generated code cannot work either. It is therefore one of the main task in the development of parser and user interface to provide the user with valueable error messages.

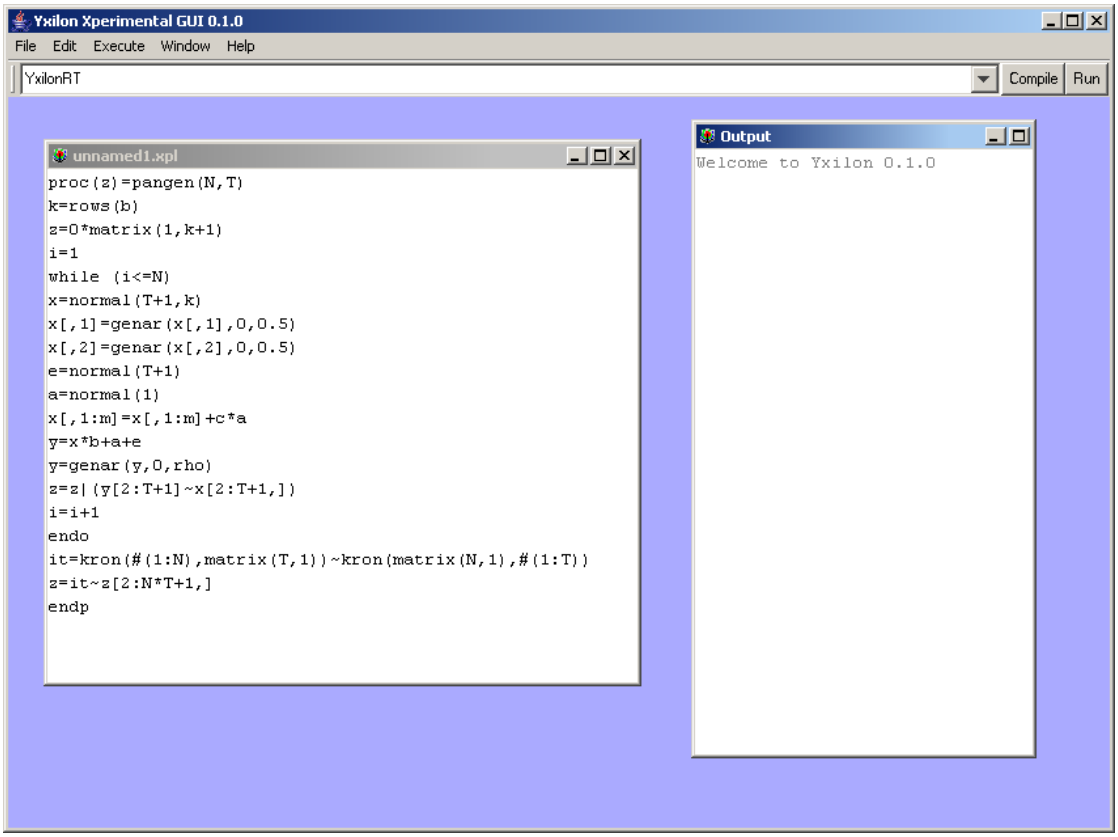


Abbildung 7.2: Yxilon Graphical User Interface

7.5 Conclusion

Yxilon is our proposal to answer the question of how future statistical packages may look like. The development of Yxilon is divided into two major stages. In the first stage we implement the parser as well as different code generators for Java and C++. For user interaction a JAVA based graphical user interface is set up allowing us to test the generated code and compilation process on different local platforms. In the second stage we extend this local approach to client/server frameworks and different databases. Different user interfaces, non-graphical and graphical as well as plugins will allow the inclusion of Yxilon methods to other office and computing environments. Source and binary distributions are updated regularly and can be found at `http://www.quantlet.org`. We are looking for feedback from users and programmers and invite them to join us in this project.

8 Yxilon - A Client/Server Based Statistical Environment (ISI 2007)

Along with many others, we agree that a modern education in statistics needs to incorporate the practical analysis of real datasets, which are usually more complex than the common examples found in standard textbooks.

The software used in the teaching of statistics includes standard spreadsheet environments such as OpenOffice and Excel and dedicated commercial and non-commercial packages such as *R*, Minitab or SPSS. With the freely available Yxilon environment we add another package and proliferate the statistical programming language XploRe, using a modern client/server based architecture. This architecture has the capabilities of serving statistical results in a variety of flavor for different groups of users. In this paper we describe the general setup of the Yxilon environment and present selected technical details.

8.1 From XploRe to Yxilon

Besides requirements such as the general stability of the software and the set of available methods, the value of a statistical software environment largely depends on the way it allows an efficient interaction with the user. Nowadays the user operates the software either via character input in a terminal or mouse. Well-known packages for the former method are *R* and Matlab, for the latter Minitab and SPSS, which also allows the extension of its methods via VisualBasic.

XploRe, the predecessor of Yxilon and its scientific fundament, belonged to the category of language-operated statistical packages. The codes, the so-called *quantlets*, were written in a C-style programming language, which was then evaluated by XploRe. XploRe includes more than 1500 quantlets, covering especially non- and semiparametric methods and the statistics of financial markets. The internal design of XploRe was very monolithic, all components were tied closely together. This made the adding of new features and the maintenance of the existing code very complicated and led to the decision to reimplement the XploRe language in the Yxilon project, using a simplified and modular client/server approach with two tiers. On the one hand this alows the easier adding of new modules, on the other hand existing modules can easily be maintained or even exchanged.

8.2 The Server and the Communication Protocol

For the communication between servers and clients there are numerous ways to exchange information. We evaluated common techniques such as *Remote Procedure Calls* (RPC) and *Remote Method Interaction* (RMI) but finally decided to develop our own binary protocol. This approach had the disadvantage that a significant amount of time had to be spent on low-level network programming but provided the advantage that we kept the complete control of the communication, which resulted in a very small overhead therefore in a very fast communication. To illustrate the communication refer to the picture below which shows an overview of the communication objects.

Currently there are ten types of objects; different types of lists to numeric and character matrices as well as a different types of exceptions. These exceptions are extremely useful when the code submitted by the user is incomplete or erroneous since they contain detailed information about the location and type of the error.

Tabelle 8.1: Overview of Communication Objects

ID	Type	Description
0	empty object	to denote `null` objects
1	numeric matrix	stores numeric matrices (up to 3 dimensions)
2	string matrix	stores string objects (up to 3 dimensions)
3	object list	may contain matrices or lists
4	named list	a named list object
5	parameter list	stores named key/value pairs
9	runtime exception	captures errors in the server
10	user exception	allows user to define own errors/warnings
13	parser exception	captures incomplete input

When Yxilon objects are sent from the server to the client as a serialized data stream, the first information transmitted is the object type. Based on this type identification the client reads the remaining bytes and generates the necessary objects as matrices and lists which are then used e.g. to generate a statistical graphics or character output.

The server component of Yxilon is currently available for Microsoft Windows only, however we will provide versions for Linux and Solaris in future. Due to the concept of modularity and exchangable parts the server itself has only very limited functionality, such as the management of objects, basic calculations and the communication stack. All computational functionality above the level of simple algebra and basic matrix manipulations has been sourced out to dynamic link libraries (DLLs). The following code listing illustrates the simple structure of these libraries. Following the initial import statements we define in the *LibMain* function a set of variables such as the author and the version of the DLL. These information, accessible also from Yxilon code, may provide valuable information e.g. in the bugtracking process.

```
#include <LibInclude.hpp>
#define CExtObj CExpCompObject

EXPORT LibMain(CExtLibrary *pLib){
  pLib->setAuthor(L"Sheldon Kelly");
  pLib->setVersion(L"1.0");
  pLib->setName(L"Mathematical Functions");
  pLib->addFunction(L"abs", L"Yabs");
  pLib->addFunction(L"ceil", L"Yceil");
  pLib->addFunction(L"floor", L"Yfloor");
  pLib->addFunction(L"cos", L"Ycos");
};
```

Listing 8.1: Example of C++ code for a dynamic link library

When the server executable is started, it reads the necessary settings from a textfile, e.g. the network port it binds to, the paths for the dlls and quantlets and the login/password needed for clients to connect. After the succesful loading of these settings a TCP/IP server port is opened. From this timepoints clients may connect.

8.3 The Client

In contrast to the server which was written in C++, the graphical user client was implemented in Sun Microsystems' Java. For our purpose Java offers verious advantages if compared with other programming languages. Compiled Java code can be run on any platform for which a Java Runtime Environment (JRE) is available, it offers a large set of methods for internet-related techniques (TCP/IP, XML, SOAP), can be embedded into HTML pages and is supported by a large user community.

The client satisfies different needs: It hides the technical implementation of the communication and offers an test platform for the implementation of new functions. The current version does not only offer basic editing and the exchange of code and results with the server but provides advanced editing features such as highlighting and autocompletion of code and database connectivity.

When the user presses the connect button, the client reads the communication settings from a textfiles and tries to exchange information, e.g. login and password. After this handshake process the user may edit his sourcefiles and send them to the server.

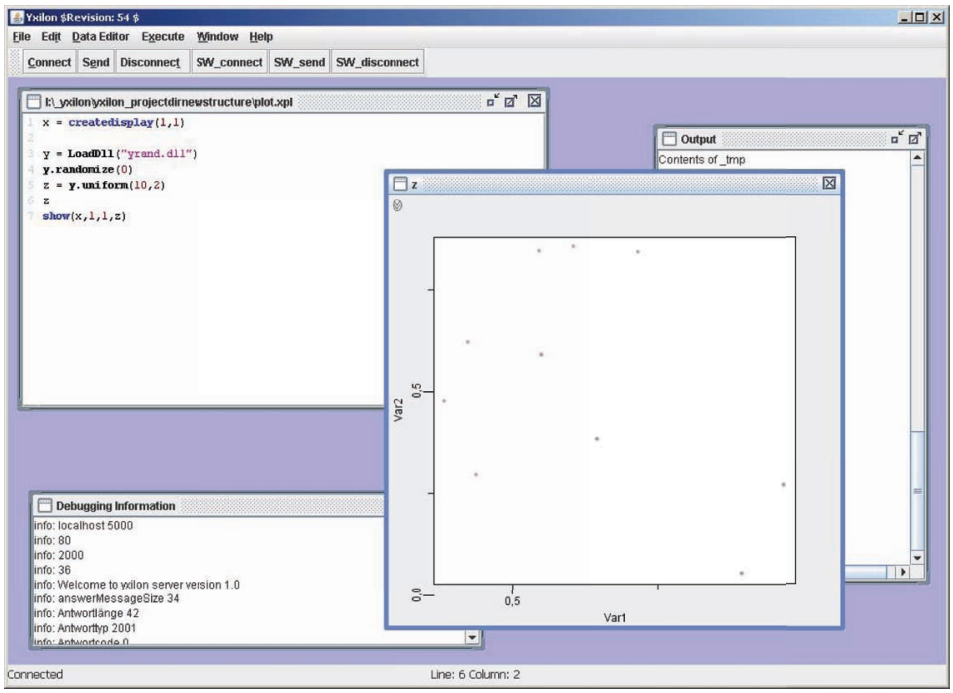


Abbildung 8.1: Yxilon Graphical User Interface

8.4 Yxilon Java client

For the display of graphics under Java there are different solutions. We have chosen the Jasplot library Nakano (2004), a library explicitly designed for the display of statistical graphics. This library can easily be embedded into Yxilon and offers a wide range of graphical means to display e.g. 2- and 3-dimensional scatterplots and parallel coordinate plots. Furthermore it allows the easy use of interactive features such as brushing and the linking of different plots.

The figure above depicts a screenshot of the Yxilon client, showing relevant windows of the interface. In the topleft corner the editor window with the Yxilon code needed to produce the scatterplot, in the bottom left the 'Debug' window, which shows selected debugging information, and the output window used for the display of numerical results. Furthermore the client also contains a database wizard, which allows to connect to different databases such as MySQL, MS Access or Oracle, via the Java Database Connector.

9 Yxilon - Description & Outlook

9.1 Introduction

While the last chapters gave an overview on selected components of the Yxilon framework, this chapter will give a brief description of the actual implementation in Java. Although the final goal, the redesign of the XploRe language in a modern environment, was not achieved, the project brought valuable knowledge and insights. The implementation of the graphical structure is used in the XploRe converter for Jasp (Fujiwara et al., 2008), which will allow to run XploRe code from Jasp (Yamamoto et al., 2007), the Yxilon Java Client in being adjusted to use other engines such as **R** (see 9.6.1).

In the next section I will describe the implemented communication protocol, based on binary TCP/IP transmission before giving an overview of the graphical client in the main section of this chapter. The chapter closes by giving an outlook, how the Yxilon Java Client can be used as an interface for **R**.

9.2 Communication Protocol

The ancestor of Yxilon, XploRe, allowed to dislocate the computing component from the user interface by the MD*Crypt protocol (Feuerhake, 2002), a Java-based communication framework based on binary TCP/IP transmissions. The server version of XploRe, the *XploRe Quantlet Server*, wrote its output to *standard out* where MD*Crypt intercepted and serialized it into a form suitable for the transmission via network. Output from the client (input for the server) was also sent to the server by *stdio* via MD*Crypt.

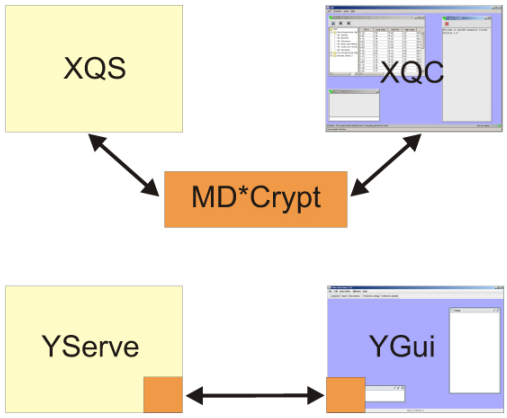


Abbildung 9.1: XploRe Quantlet Server and Yxilon Server communication architecture

The drawback of this solution was that the additional communication layer via standard input and output resulted in a slow overall communication. For larger datasets Yxilon was supposed to work with this seemed impractical.

A significant increase in the communication speed was therefore one of the main goals in the Yxilon project. Since the server application was developed in C++ while the client application was implemented using SUN Java, the solution had to be independent from architectures limited to one platform.

For an overview of both architectures see Figure 9.1. In the following the communication protocol, as it had been developed by the author and Yuval Guri, the student assistant, is described in detail.

A well-known annoyance of many applications, regardless in which programming language they have been implemented, is the problem of lacking response when computationally or network intensive tasks are executed. The reason for this behaviour lies in the way the frameworks used to develop graphical user interfaces are handling the events from the application.

The Java Abstract Windowing Toolkit (AWT) uses a single-threaded model for painting the graphical components, all updates of those screen components are performed from a single thread. If now long-running functions are called by clicking a button or menu item, this updating thread has to wait until the end of the function thus blocking the use of the user interface.

To avoid these locking effects in the Yxilon client we had to source out all activities which might block the interface to threads apart from the single event thread by using swingworker instances.

Swingworker is the name of a utility class by SUN Microsystems which allows the execution of time-consuming tasks in the background, leaving the user interface ready for input. Swingworker instances are used in various parts of the Yxilon client, mainly in function which need information or data from the network.

An example is shown in Listing 9.1, showing the method that is called when the user presses the `connect` button of the client. The methods `CommHandler2.handshake()` and `CommHandler2.handshake2()` execute the initial opening of the communication between server and client. If both methods are executed successfully they return `true` and the client can start the communication. Via LabelManager (see 9.3) the client indicates a successful connect and transmits the contents of the `startup.xpl` file to the server. The results of this transmission are then sent to the output window of the client. To supervise the communication status between server and client we use another swingworker instance, implemented in `Commhandler4.java`, see below for details.

```
1  connect.addActionListener(new java.awt.event.ActionListener() {
2   public void actionPerformed(java.awt.event.ActionEvent evt) {
3    try {
4     // handshake methods
5     boolean success = CommHandler2.handshake(pp);
6     boolean success2 = CommHandler2.handshake2(pp);
7
8     if (success & success2){
9      lp.info("Both sockets are connected");
10     lm.setLeft("Connected");
11     send.setEnabled(true);
12     String startup = "startup.xpl not found";
13     startup = GuiTools.loadFile(new File("startup.xpl"));
14     CommHandler2.sendServer(startup);
15     CommHandler2.getServer(gui.output);
16     new ConnWorker4().execute();
17    }
18   } catch (java.lang.NullPointerException n){
19     gui.debug.append(n.toString()+"\n");
20     lp.info(n.toString());
21    }
22    catch (Exception e){
23     lp.info(e.toString());
24       }
25       }
26     });
```

Listing 9.1: Yxilon Client: Connect method from `GuiToolbar.java`

Listing 9.3 shows the `handshake()` method which is called in lines 5 and 6 of Listing 9.1. This method is used to establish the initial communication between server and client.

After setting up the instance of the `LoggerProvider`, input and output connections are opened. The server then sends a message asking for login and password string to authentificate the client. Following this authentification the server sends the greeting message ('Welcome to Yxilon') and a session ID to differentiate between simultaneously connected clients. At this point the handshake is complete, the communication is set up for the transmission of code and data.

An overview of the communication steps required for the handshake is shown in Table 9.1.

Tabelle 9.1: Communication Flow during Handshake

Server		Client	Description
messageLength	⇒		overall length of the message
messageType	⇒		message type
stringLength	⇒		length of welcome string
stringLength	⇒		length of welcome string
welcomeString (sb)	⇒		request for login/password
	⇐	answerMessageSize	overall length of the answer
	⇐	YCM_LOGIN	login type
	⇐	login	login name
	⇐	password	login password
answerLength	⇒		length of the answer
answertype	⇒		type of answer
answerCode	⇒		message type
answerstringlength	⇒		length of the answer from server
answerstringlength (sb)	⇒		answer text
SessionID	⇒		sessionID

```
1  public static boolean handshake(Properties p) throws Exception {
2    // Instantiation of the logging environment
3    LoggerProvider lp = LoggerProvider.getInstance();
4    // reading variables from property bundle
5    String host = p.getProperty("host");
6    int port = new Integer(p.getProperty("port")).intValue();
7    int timeout = new Integer(p.getProperty("timeout")).intValue();
8
9    lp.info(host + " " + port);
10   String login = p.getProperty("user");
11   String password = p.getProperty("password");
12   // open socket
13   dataSocket = new Socket(host, port);
14   dataSocket.setSoTimeout(timeout);
15   dataSocket.setKeepAlive(true);
16   // open input/output streams
17   dataDataInputStream = new DataInputStream(dataSocket.
       getInputStream());
18   dataDataOutputStream = new DataOutputStream(dataSocket.
       getOutputStream());
19   Thread.sleep(200);
20   // reading from server
21   int messageLength = dataDataInputStream.readInt();
22   lp.info(messageLength);
23   short messageType = dataDataInputStream.readShort();
24   lp.info(messageType);
```

```
25    short stringLength = dataDataInputStream.readShort();
26    lp.info(stringLength);
27    StringBuffer sb = new StringBuffer(stringLength);
28    short input;
29     for (int i=0;i<stringLength;i++){
30       sb.append((char)dataDataInputStream.readShort());
31     }
32
33    lp.info(sb.toString()); // output
34
35    int answerMessageSize = login.length()*2+password.length()*2+10;
36    lp.info("answerMessageSize " + answerMessageSize);
37
38    dataDataOutputStream.writeInt(answerMessageSize);
39    dataDataOutputStream.writeShort(YCM_LOGIN);
40    writeStream(dataDataOutputStream, login);
41    writeStream(dataDataOutputStream, password);
42
43    int answerlength = dataDataInputStream.readInt();
44    short answertype = dataDataInputStream.readShort();
45    short answerCode = dataDataInputStream.readShort();
46    short answerstringlength = dataDataInputStream.readShort();
47
48    lp.info("Answer length "+answerlength);
49    lp.info("Answer type "+answertype);
50    lp.info("Answer code "+answerCode);
51
52    sb = new StringBuffer(answerstringlength);
53     lp.info("Reading from Socket...1");
54          for (int i=0;i<answerstringlength;i++){
55             sb.append((char)dataDataInputStream.readShort());
56          }
57    lp.info(sb.toString());
58
59    SessionID = dataDataInputStream.readInt();
60    lp.info("SessionID: " + SessionID); // output
61
62    if (answerCode==1) {
63     return false;
64    }
65     return true;
66    }
```

Listing 9.2: Yxilon Client: `handshake` method from `Commhandler2.java`

Listing 9.3 shows an interesting use of the swingworker class. This `ConnWorker4` called class is used to watch the data and code communication between server and client. Every second an instance of this class sends a background message to the server to check its status. Based on the result of this operation the graphical user interface sets the label in the right bottom corner, indicating the status of the communication.

As this method runs as long as the connection is active, there is no need to fill the done() method which is usually called after the background task has been finished.

```
class ConnWorker4 extends javax.swing.SwingWorker<Boolean, Object>{
  java.util.Properties pp;
  LoggerProvider lp;
  LabelManager lm;

  public ConnWorker4(){
    lp = LoggerProvider.getInstance();
    lm = LabelManager.getInstance();
  }

  @Override public Boolean doInBackground(){
    try {
      Thread.sleep(1000);
      int i=0;
      while (true){
        CommHandler2.sendStatus();
        short status = CommHandler2.getStatus();
        lm.setRight("Status_" + i + ": " + status);
        Thread.sleep(1000);
        i++;
       }
    } catch (Exception e){
      lp.warn(e.toString());
    }
      return true;
  }

  @Override
    protected void done()
    {
      try {

      }
      catch (Exception e ) { }
    }
  }
```

Listing 9.3: Yxilon Client: `Commhandler4.java`

In the following we will go on with the description of the communication and focus on the transmission of objects from the client to the server and vice versa. Listing 9.4 shows the `sendserver()` method from the `Commhandler2` class, which is used to send string matrices to the server. After setting up the instance of the log4j logging framework the name of the matrix that is to be transmitted is converted from a string sequence to a character array, simplifying the following transmission operations. The size of the message is then calculated as following:

messagesize an integer with four byte plus

messagetype as short with two byte plus

length of the name , short with two byte plus

name one short value (two bytes) for each character plus

layers of the matrix one integer with four bytes plus

number of rows and columns two integers with eight bytes plus

length of string & string itself for each string in the matrix one short is sent for the length and for each string character one short.

By using this method of communication the communication overhead, the amount of data which has no direct information value, can be kept at a very small level since only a few byte need to be transferred in addition.

```
1  public static void sendServer(String[][] data, String name)
2  throws Exception{
3
4    LoggerProvider lp = LoggerProvider.getInstance();
5    char[] nameArray = name.toCharArray();
6    short nameLength = (short)nameArray.length;
7    int rows = data.length;
8    int cols = data[0].length;
9    int size = 4 + 2 + 2 + nameLength * 2 + 12 + (rows * cols) * 4;
10   lp.info("Sending string matrix " + size);
11   dataDataOutputStream.writeInt(size);
12   dataDataOutputStream.writeShort(1008);
13   dataDataOutputStream.writeShort(nameLength);
14
15   for (int j=0;j<nameLength;j++){
16     dataDataOutputStream.writeShort(Character.codePointAt(nameArray,j)
         );
17   }
18
19   dataDataOutputStream.writeInt(1);
20   dataDataOutputStream.writeInt(rows);
21   dataDataOutputStream.writeInt(cols);
22   for (int r=0;r<data.length;r++){
23     for (int c=0;c<data[0].length;c++){
24       dataDataOutputStream.writeShort(1);
25       char[] temp = data[r][c].toCharArray();;
26       dataDataOutputStream.writeShort(Character.codePointAt(temp,0));
27     }
28   }
29 }
```

Listing 9.4: `sendServer()` method from `Commhandler2.java`

Listing 9.5 shows an excerpt of the `getServer()` method which together with the `getObject()` method from Listing 9.6 is used to receive objects. After the obligatory setup of the logging instance the method first reads the overall length of the message to be received from the server. After receiving the type of the answer (results, errors, etc.) and further parameters the `getObject()` method is called, taking as parameter the number of objects to be received and the link to the output window, where the content of the objects will be presented to the user.

```
1  public static short getServer(GuiOutputFrame output, GuiDebugFrame
       debug) throws Exception{
2    LoggerProvider lp = LoggerProvider.getInstance();
3    int answerlength = dataDataInputStream.readInt();
4    short answertype = dataDataInputStream.readShort();
5    short eom = dataDataInputStream.readShort();
6    short objCount = dataDataInputStream.readShort();
7    lp.info("getserver_answerlength: " + answerlength);
8    lp.info("getserver_answertype: " + answertype);
9    lp.info("getserver_eom: " + eom);
10   lp.info("getserver_objCount: " + objCount);
11   getObject(objCount, output,"");
12   return eom;
13 }
```

Listing 9.5: `getServer()` method from `Commhandler2.java`

Listing 9.6 shows the `getObject()` method which is used to distinguish between the different object types available in Yxilon. Supported are

- numeric matrices
- string matrices
- named and unnamed lists
- various types of exceptions

For each object type this method implements the necessary handling functions to show the object data in the output window.

```
1  public static void getObject(int objectCount, GuiOutputFrame output,
       String listname) throws Exception{
2    LoggerProvider lp = LoggerProvider.getInstance();
3    lp.info("receiving " + objectCount + " Object(s)");
4
5    for (int objCounter=0;objCounter<objectCount;objCounter++){
6      short type = dataDataInputStream.readShort();
7      lp.info("object type " + type);
```

```
8     lp.info(">object name " + listname);
9
10    short objectnamelength = dataDataInputStream.readShort();
11    lp.info("length of object name " + objectnamelength);
12    StringBuffer sb = new StringBuffer(objectnamelength);
13    for (int i=0;i<objectnamelength;i++){
14      sb.append((char)dataDataInputStream.readShort());
15   }
16
17    switch (type){
18      case 1: { // numeric matrix
19        output.append("numeric matrix\n");
20        lp.info("receiving numeric matrix");
21        int rows = dataDataInputStream.readInt();
22        int cols = dataDataInputStream.readInt();
23        int layers = dataDataInputStream.readInt();
24        lp.info("rows " + rows);
25        lp.info("cols " + cols);
26        lp.info("layers " + layers);
27        int dim = cols * rows * layers;
28        double x[] = new double[dim];
29        for (int counter=0; counter<dim; counter++){
30        x[counter] = dataDataInputStream.readDouble();
31      }
32
33        int a = -1;
34        double[][][] sp = new double[layers][rows][cols];
35        for (int j=0;j<layers;j++){
36          for (int m=0;m<rows;m++){
37            for (int l=0;l<cols;l++){
38              a++;
39              sp[j][m][l]=x[a];
40         }}}
41         format(output, sp);
42    }
43    break;
44
45    case 2: { // String Matrix
46      lp.info("receiving string matrix");
47      output.append("string matrix");
48      int rows = dataDataInputStream.readInt();
49      int cols = dataDataInputStream.readInt();
50      int layers = dataDataInputStream.readInt();
51      lp.info("rows " + rows);
52      lp.info("cols " + cols);
53      lp.info("layers " + layers);
54      int dim = cols * rows * layers;
55      String x[] = new String[dim];
56
```

```
57      for (int counter=0; counter<dim; counter++){
58        short stringLength = dataDataInputStream.readShort();
59        sb = new StringBuffer(stringLength);
60          for (int i=0;i<stringLength;i++){
61            sb.append((char)dataDataInputStream.readShort());
62          }
63         x[counter] = "\"" + sb.toString() + "\"";
64      }
65      int a = -1;
66      String[][][] sp = new String[layers][rows][cols];
67      for (int j=0;j<layers;j++){
68        for (int m=0;m<rows;m++){
69          for (int l=0;l<cols;l++){
70            a++;
71            sp[j][m][l]=x[a];
72      }}}
73      format(output, sp);
74    }
75    break;
76
77    [...]
78
79    } // end switch
80  }
81 }
```

Listing 9.6: Excerpt from `getObject()` method from Commhandler2.java

9.3 Yxilon Java Client (YJC)

As the general type of the software has not changed there are similarities to the interface of the XploRe standaloneversion, especially with respect to the design and layout of the windows, and differences since buttons and toolbars for the communication needed to be integrated. Figure 9.3 shows a screenshot of the current Yxilon client version, showing the input, output window with a small code sample.

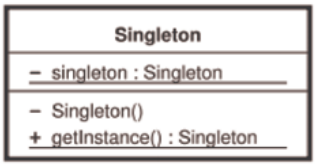


Abbildung 9.2: Singleton pattern in UML (Wikipedia, 2008b)

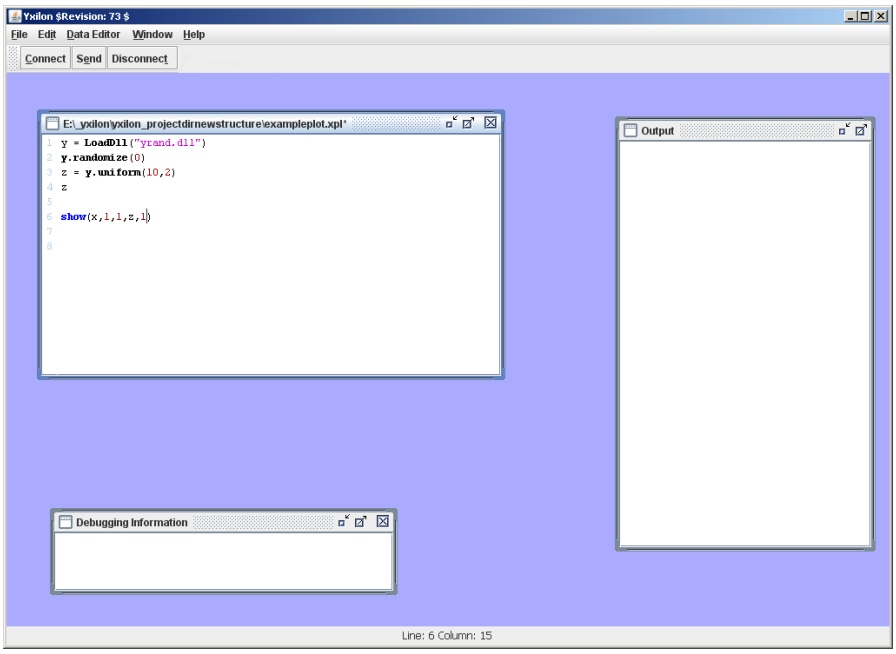


Abbildung 9.3: Yxilon Java Client

The implementation of the client uses so-called 'Design Patterns' in various places therefore we will give a brief overview on this technique. 'Design Patterns' (Freeman et al., 2004) are reusable solutions to frequently occuring problems in object-oriented software development. They are usually grouped into patterns for the creation, structure, behavior and concurrency of objects (Wikipedia, 2008a). Other classifications use architectural approaches applied at the architecture level of the software, well known is the Model-View-Controller pattern.

The pattern most frequently used in Yxilon is the Singleton pattern, a pattern to make sure that one and only one instance of a certain class is instantiated and accessible program wide. Figure 9.2 shows the UML representation of singletons.
Listing 9.7 shows how a singleton pattern is used inside the client. In the bottom menubar of the Yxilon client there are various labels indicating line and column in the editor as well as the connection status. As these labels are updated by various events from different classes the use of a singleton allows a more elegant design as references to the label manager do not need to be transferred to the class instances from where changes of the labels might be called.

The `LabelManager` class uses four variables, three labels for the left, center and right label and one instance of itself, the `Labelmanager lm`. The access of the constructor, the method which is called when a new instance of the class is instantiated, is set to private so only the `LabelManager` itself could create new instances. The globally accessible `getInstance()` is the method called from other classes and controlles the creation of the class. If no `LabelManager` has been instantiated so far a new instance is created and returned to the calling class, otherwise a reference to the previously created

instance is returned.

```
1  public class LabelManager {
2     private static LabelManager lm;
3     private static javax.swing.JLabel left = new javax.swing.JLabel("
          );
4     private static javax.swing.JLabel center = new javax.swing.JLabel(
          "Line: 0 Column: 0");
5     private static javax.swing.JLabel right = new javax.swing.JLabel("
          ");
6
7     private LabelManager(){ }
8
9     public static LabelManager getInstance(){
10       if (lm==null){
11          lm = new LabelManager();
12          return lm;
13       } else { return lm; }
14    }
15
16    public void setLeft(String s){
17       left.setText(s);
18    }
19
20    [...]
21
22    public javax.swing.JLabel getLeftLabel(){
23       return left;
24    }
25
26    [...]
27 }
```

Listing 9.7: Excerpt from `LabelManager.java`

The advantage of design patterns lies in their universality and dispersion. For object-oriented software development they are the standard to solve recurrent problems, most people dealing with software development in Java or C++ know about their existence in use thus simplifying the process of dealing with software written by other programmers and finding a common lingo in the discussion.

9.4 Database connection

With the ability to generate and store huge amounts of data, e.g. orderbook tick data, efficient ways need to be found to process and evaluate these data are needed. The storage in a relational database system like Oracle, SQL Server or MySQL provides

efficient ways to run various data aggregation and selection procedures, however their functionality with respect to statistical functions is rather limited.

With JDBC (Java Database Connectivity) there exists a general and powerful connector framework, comparable to ODBC on Windows, by which databases can be accessed from Java. For Yxilon an import wizard has been developed which conviently allows the user to select datasets from database systems and to import them to Yxilon.

In the first dialogue database host, login and password are specified. The JDBC driver is specified by selecting an item from the list. New list items can be specified by adding entries for driver, host, login and password in the respective ini file.

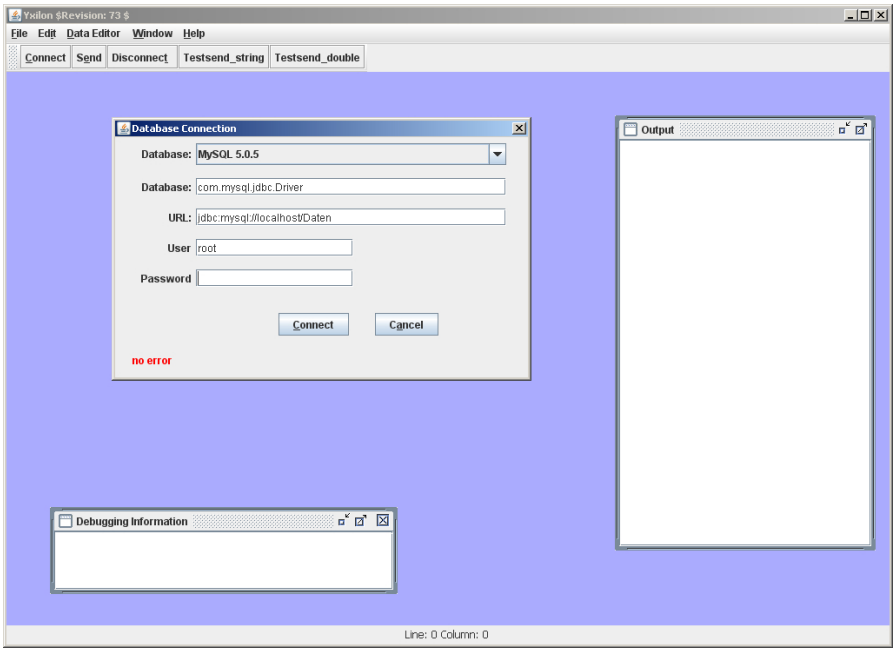


Abbildung 9.4: Database wizard: Connector selection

The second dialogue allows the user to specify the SQL query. Furthermore a column separator can be specified, which is used to separate the different columns in the result-set.

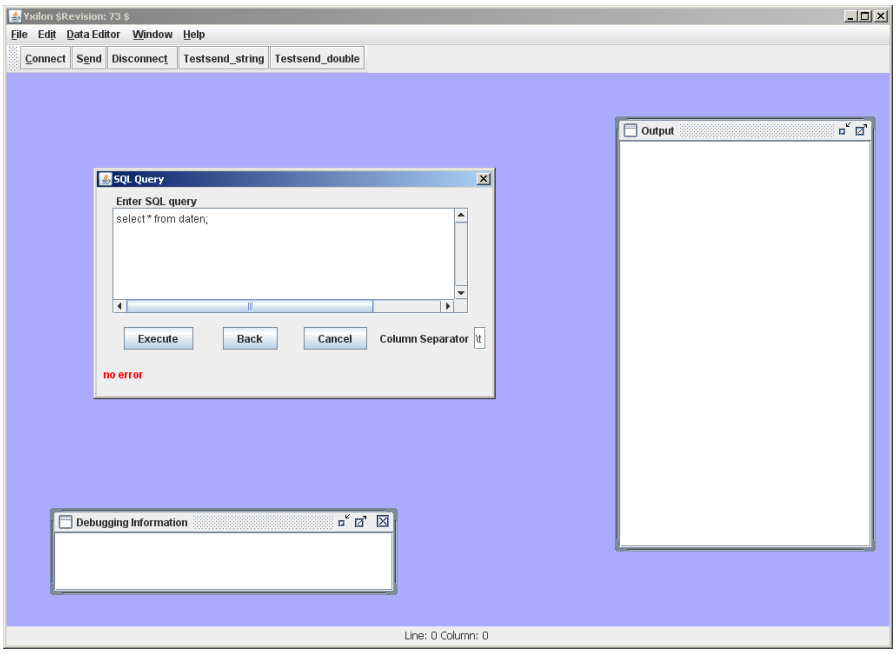


Abbildung 9.5: Database wizard: Specification of the SQL query

In the third dialogue the wizard displays the retrieved data in a text window. The user may now copy the data to the clipboard, save it in a textfile or send it directly to the matrix editor of the user client, see Figure 9.7.

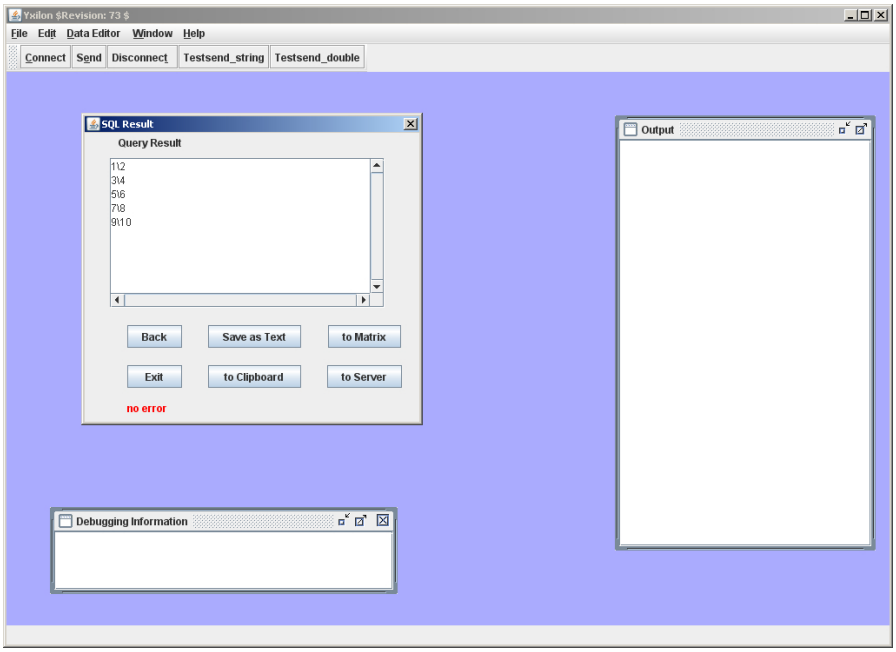


Abbildung 9.6: Database wizard: Result of the query

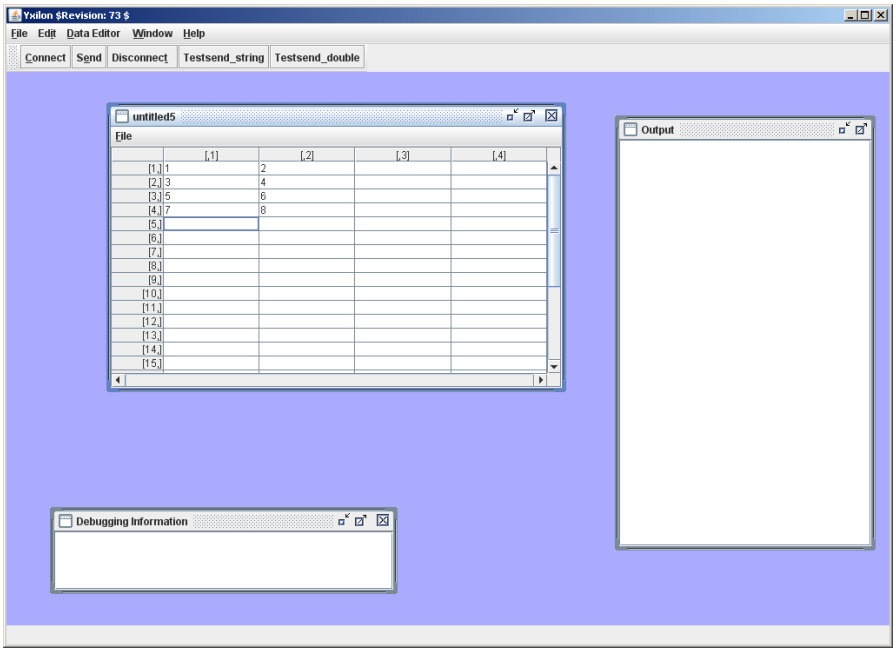


Abbildung 9.7: Database wizard: Query result in the matrix editor

9.5 Graphics Implementation

9.5.1 Introduction

The ability to display graphics is a necessary key feature of each statistical software package since humans are better in judging pictures than in judging tables of numbers. In an experiment described by Stock and Watson (1984) two groups participants had to classify companies according to given financial information. While the first group received just the plain numbers, the second group received charts with Chernoff-faces (Chernoff, 1973; Flury and Riedwyl, 1981), representing the numbers in graphical form. The second group made significantly better judgements.

The question of usefulness of statistical charts needs no further discussion however we need to discuss which features a graphics engine needs to be equipped with to be useful in data analysis. Although some of these requirements have been mentioned in earlier chapters we will describe them here to give a short overview.

Probably the most important feature is the provision of a standard set of graphics, graphics that are wellknown and can easily be interpreted. While the list in this set may of course vary, the following chart types can be seen as a common standard, implemented by all statistics packages:

- Scatterplot
- Box-Whisker plot

- Histogram
- Line chart
- Starplot

Depending on the type of analysis the user may or may not need a certain level of interactivity. For exploratory data analysis the use brushing, highlighting and linking may provide valuable information in exploring the data (Unwin et al., 2006), for books and reports this may not be of great importance.

In the analysis of complex datasets additional factors such the ability to efficiently deal with thousand and millions of datapoints play an important role since standard graphics may be of no use in these cases. While the computer may be very fast, the plotting of 2-3 million data points in a common scatterplot is very likely to produce no valuable results, other plot types such as described in Unwin et al. (2006) are to be preferred then. Hardware-accelerated graphics (DirectX, OpenGL) may help but in the case of millions of points even these techniques may prove to be much too slow.

A final requirement for a statistical graphics engine is the ability to support new graphics by allowing the user to develop new types of charts either from graphical primitives such as lines, boxes and circles or the extension of existing graphs.

For Yxilon this requirement was closely connected with being compatible to the existing graphics model used in XploRe 4. The XploRe graphics model is based on list objects containing information on the attributes of points and lines, such as size, color and style. Listing 9.8 shows basic XploRe code to generate a box-whisker plot.

```
1 library("graphic")
2 d = createdisplay(1,1)
3 p = grbox(normal(100,1))
4 show(d,1,1,p)
```

Listing 9.8: XploRe code to generate a simple boxplot

When we look at the p list object, we find various matrices inside:

p.data $x - y$ coordinates of the points to be plotted

p.lines information on which points are to be connected by lines

p.lcolor information on the line color

p.lstyle line styled

p.lsize line thicknesses

p.pcolor colors of points

p.pstyle point styles

p.psize point sizes

In the following we will describe the necessary adjustments made to Jasplot to be able to handle XploRe graphical objects.

9.5.2 Implementation of XploRe Graphics Structure in Jasp

The design and implementation of our own graphics engine would have been too time-consuming, so we decided to integrate and adapt Jasplot according to our needs, a statistical graphics package developed in Java. As this package, developed at the Institute of Statistical Mathematics (ISM) in Tokyo, has a structure which is different from the XploRe graphics structure, significant changes had to be made.

Listing 9.9 shows an example source code from the Jasp developers which produces Figure 9.8. Based on this code and the graphics we will explain the basic structure of Jasplot.

Datasets are managed by Jasplot using so-called data models, which provide the necessary functionality to access textfiles (`CSVDataModel`), Excel (`ExcelDataModel`) or tree structures (`TreeDataModel`). The information from this data model is then used to set up the model for the plot (in this case the scatterplot model, an instance of `ScatterPlotModel.java`), which collects the information on symbol sizes, line parameters and connection groups (groups of connected points). Internally this class calls the `ScatterPlotter` class, which does the actual painting on the canvas. The generated `Graphics` respectively `Graphics2D` object is then painted on the `JasplotPanel` instance and shown to the user.

These `xxxPlotModel` and `xxxPlotter` classes have been implemented for each graphics type in Jasplot. While this allows a very fine adjustment of each chart type it requires a significant amount of time to implement new charts for Jasplot in Java directly. The pnuts (Tomatsu, 2008) language used to write the Jasp code, cannot be used. With the extension of Jasplot by the XploRe graphics model we add this flexibility, as new statistical graphics types may be created by simply generating the corresponding arrays for lines, points and their behavior; the underlying Java codebase does not need to be modified.

```
1  package ScatterPlot;
2  import javax.swing.JFrame;
3  import jp.jasp.jasplot.CSVDataModel;
4  import jp.jasp.jasplot.DataModel;
5  import jp.jasp.jasplot.JasplotPanel;
6  import jp.jasp.jasplot.ScatterPlotModel;
7
8  public class Sample {
9    public Sample() {
10     // load iris data
11     DataModel dataModel = new CSVDataModel("data/iris2.csv");
12     // generate scatterplotmodel
```

```
13      ScatterPlotModel model = new ScatterPlotModel(dataModel, 1, 2);
14      // create Jasplot panel
15      JasplotPanel jasplot = new JasplotPanel(model);
16      // set up Java window
17      JFrame jFrame = new JFrame("Jasplot sample: Scatterplot");
18      jFrame.getContentPane().add(jasplot);
19      jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20      jFrame.setSize(500, 500);
21      jFrame.setVisible(true);
22    }
23
24    public static void main(String[] args) {
25        Sample sample = new Sample();
26    }
27  }
```

Listing 9.9: `Sample.java`, the Jasplot scatterplot example

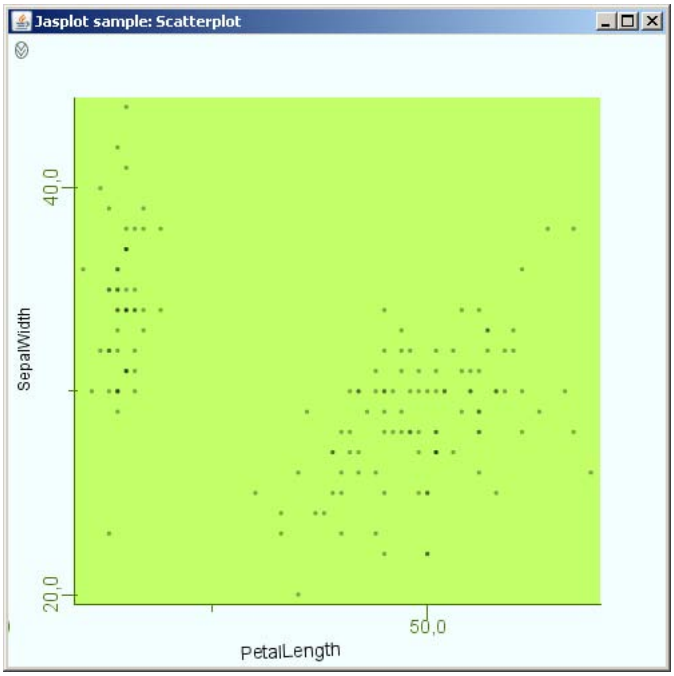


Abbildung 9.8: Jasplot Scatterplot

The following code from Listing 9.10 shows, together with the resulting Figure 9.9, the available plot symbols, colors and lines which can be used. The example uses a 5×5 array of points, which are fed into a `GenericDataModel`. The datamodel is then used to set up the `XploRePlotModel`, after setting the symbol, size and line parameters the plot is created.

```
1  package XploRePlot;
2  import java.util.ArrayList;
3  import javax.swing.JFrame;
4  import jp.jasp.jasplot.CSVDataModel;
5  import jp.jasp.jasplot.DataModel;
6  import jp.jasp.jasplot.JasplotPanel;
7  import jp.jasp.jasplot.ScatterPlotModel;
8  import jp.jasp.jasplot.*;
9  import java.awt.*;
10 import java.awt.BasicStroke;
11
12 import java.awt.Color;
13
14 /**
15  * example to show the features of the XploRe plot implementation
16  * @author Uwe Ziegenhagen
17  */
18 public class AllSample {
19   // converts an array of linewidth into an arraylist
20   public static ArrayList generateLineWidths(double linew[]){
21     ArrayList al = new ArrayList();
22     for (int i = 0; i<linew.length;i++){
23       al.add(linew[i]);
24     }
25     return al;
26   }
27
28   public static void main(String[] args){
29     // we use a 5x5 array as data matrix
30     double[][] data = new double[][]{{1,1},{1,2},{1,3},{1,4},
31     {1,5},{2,1},{2,2},{2,3},{2,4},{2,5},{3,1},{3,2},{3,3},{3,4},
32     {3,5}, {4,1},{4,2},{4,3},{4,4},{4,5},{5,1},{5,2},{5,3},
33     {5,4},{5,5}};
34
35     // now we prepare the arraylist for the data model
36     ArrayList al = new ArrayList();
37     ArrayList x = new ArrayList();
38     ArrayList y = new ArrayList();
39
40     for (int i = 0;i<data.length;i++){
41       x.add(data[i][0]);
42       y.add(data[i][1]);
43     }
44
45     al.add(x);
46     al.add(y);
47     // generate datamodel
48     GenericDataModel generic = new GenericDataModel(al);
49     XploRePlotModel model = new XploRePlotModel(generic);
```

```
50
51      // plot area
52      model.setFieldColor(Color.WHITE);
53      // area outside plot area
54      model.setBackgroundColor(Color.WHITE);
55
56      // now we set the colors of the points
57      int pcolor[] = new int[]{0,0,0,0,0,0,0,0,0,0,0,
58      0,0,0,0,0,0,0,0,0,1,0,0,0,1};
59      model.setPointColor(pcolor);
60
61      // the color of the lines and the line definitions
62      // connect 1st with 2nd, 2nd with 3rd etc.
63      int[][] lines = new int[][]{{1,2},{2,3},{3,4},
64      {4,5},{5,6},{6,7},{7,8}};
65
66      // increasing width of lines
67      int[] lineWidth = new int[]{1,2,3,4,5,6,7};
68      // style of the lines, see
69      int[] lineStyle = new int[]{1,2,3,4,5,6,7};
70      int[] lineColor = new int[]{0,1,2,3,4,5,6};
71
72      model.setLines(lines);
73      model.setLineWidths(lineWidth);
74      model.setLineStyles(lineStyle);
75      model.setLineColors(lineColor);
76
77      // using different point styles
78      int[] pstyle = new int[]{1,2,3,4,5,6,7,8,9,
79      10,11,12,13,14,1,2,3,4,5,6,7,8,9,10,15};
80      model.setConnectionGroupsSymbol(pstyle);
81      // if no connectionGroupsSymbols are given, use default symbol
82
83      // size of the points, default size is 3. Note that XploRe itself
            defines only sizes from 0 to 15
84      int[] psize = new int[]{1,2,3,4,25,6,7,8,9,10,11,12,13,14,15,16,
            17,18,19,20,21,22,23,24,25};
85      model.setPointSize(psize);
86
87      // setting up the window and generating the plot
88      JasplotPanel jasplot = new JasplotPanel(model);
89      JFrame jFrame = new JFrame("Jasplot sample: XplorePlotter");
90      jFrame.getContentPane().add(jasplot);
91      jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
92      jFrame.setSize(500, 500);
93      jFrame.setVisible(true);
94      }
95  }
```

Listing 9.10: `AllSample.java` summarizing the XploRePlotModel in Jasp

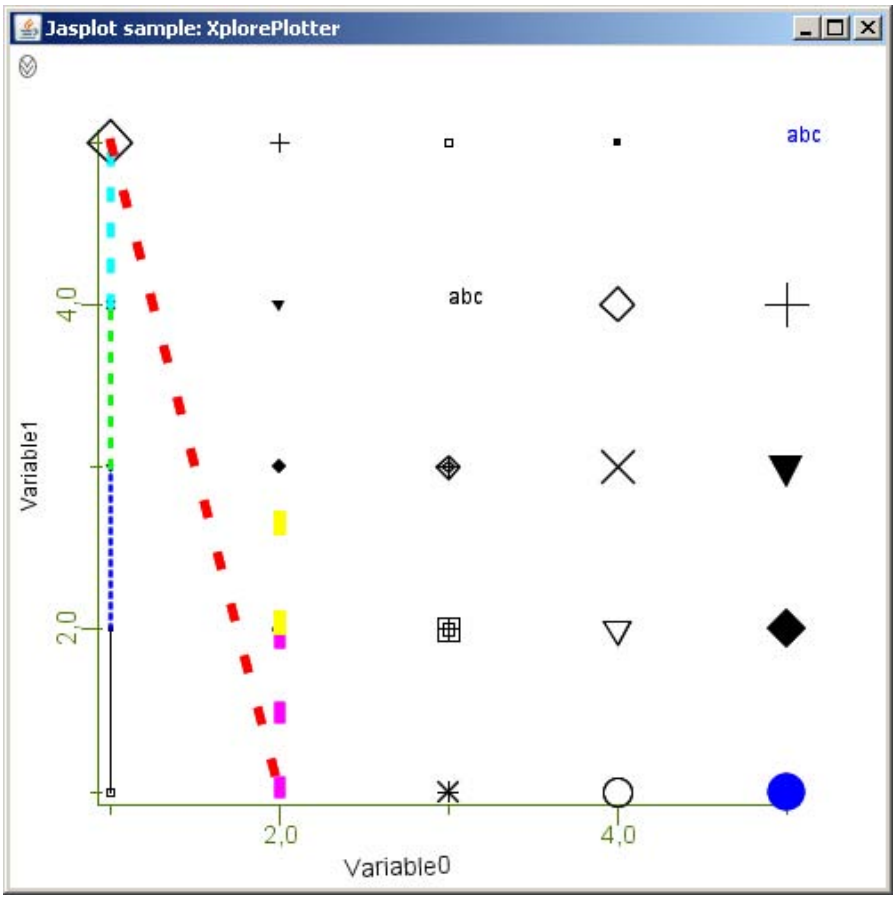


Abbildung 9.9: Output of `AllSample.java`

The following two listings show the implementation of `XploRePlotModel.java` (Listing 9.11) and `XploRePlotter.java` (Listing 9.12 on page 92). These classes implement the necessary changes to make Jasplot compatible with the XploRe graphics structure.

```
1  // by U. Ziegenhagen and JASP team
2  package jp.jasp.jasplot;
3  import java.awt.Color;
4  import java.io.IOException;
5  import java.io.ObjectInputStream;
6  import java.io.ObjectOutputStream;
7  import java.util.ArrayList;
8  import jp.jasp.jasplot.util.symbol.*;
9
10 public class XploRePlotModel extends BasicPlotModel {
11    static final long serialVersionUID = -8431852058810265798L;
12    private int[][] lines = null;
```

```
13    private int[] lineWidths = null;
14    private int[] lineStyles = null;
15    private int[] lineColors = null;
16
17    public void setLines(int[][] l){
18       this.lines = l;
19    }
20
21    public void setLineWidths(int[] l){
22       this.lineWidths = l;
23    }
24
25    public void setLineStyles(int[] s){
26       this.lineStyles = s;
27    }
28
29    public void setLineColors(int[] c){
30       this.lineColors = c;
31    }
32
33    public int[][] getLines(){
34       return lines;
35    }
36
37    public int[] getLineWidths(){
38       return lineWidths;
39    }
40
41    public int[] getLineStyles(){
42       return lineStyles;
43    }
44
45    public int[] getLineColors(){
46       return lineColors;
47    }
48
49
50
51    public static ArrayList<Symbol> generateSymbols(int pstyle[]){
52       ArrayList<Symbol> al = new ArrayList<Symbol>();
53
54       for (int i = 0; i<pstyle.length;i++){
55          switch (pstyle[i]){
56             case 0: al.add(new Empty());break;
57             case 1: al.add(new Square(false));break;
58             case 2: al.add(new Circle(false));break;
59             case 3: al.add(new XploReTriangle(false));break;
60             case 4: al.add(new Xsymbol());break;
61             case 5: al.add(new Rhombus(false));break;
```

```
62              case 6: al.add(new Square(true));break;
63              case 7: al.add(new Circle(true));break;
64              case 8: al.add(new Rhombus(true));break;
65              case 9: al.add(new XploReTriangle(true));break;
66              case 10: al.add(new Cross());break;
67              case 11: al.add(new XploReStar());break;
68              case 12: al.add(new XploReSquare());break;
69              case 13: al.add(new XploReRhombus());break;
70              default: al.add(new TextSymbol("abc"));
71          }
72       }
73       return al;
74    }
75
76       public static java.util.ArrayList generateColors(int colors[]){
77       java.util.ArrayList al = new java.util.ArrayList();
78       for (int i = 0; i<colors.length;i++){
79          switch (colors[i]){
80             case 0: al.add(Color.BLACK);
81             break;
82             case 1: al.add(Color.BLUE);
83             break;
84             case 2: al.add(Color.GREEN);
85             break;
86             case 3: al.add(Color.CYAN);
87             break;
88             case 4: al.add(Color.RED);
89             break;
90             case 5: al.add(Color.MAGENTA);
91             break;
92             case 6: al.add(Color.YELLOW);
93             break;
94             case 7: al.add(Color.WHITE);
95             break;
96             default: al.add(Color.BLACK);
97          }
98       }
99       return al;
100   }
101   [...]
102
103
104   /**
105    * Creates XploRePlotModel with the DataModel.
106    * DataModel specified here serves as a default data of the graph
107    * displayed.
108    *
109    * @param model DataModel for displaying as a default data model.
110    */
```

```
111     public XploRePlotModel(DataModel model) {
112        this(model, 0, 1);
113        this.setDefaultSymbol(new Circle(false));
114     }
115
116     public XploRePlotModel() {
117        super();
118        this.setDefaultSymbol(new Circle(false));
119     }
120
121     public Plotter createPlotter() {
122        return new XploRePlotter(this);
123     }
124
125     /**
126      * Sets a rule for row of line draw.
127      * @param row is the ArryList of ArrayList.
128      */
129     public void setConnectionGroups(ArrayList<ArrayList> row) {
           connectionGroups = row; }
130
131     /**
132      * Sets a rule for row of line draw.
133      * @param groups is the array of array.
134      */
135     public void setConnectionGroups(int groups[][]) {
136        ArrayList<ArrayList> row = new ArrayList<ArrayList>();
137        for(int i = 0; i < groups.length ; i++) {
138           ArrayList<Integer> gl = new ArrayList<Integer>();
139           for(int j = 0 ; j < groups[i].length ; j++) {
140              gl.add(new Integer(groups[i][j]));
141           }
142           row.add(gl);
143        }
144        setConnectionGroups(row);
145     }
146
147     /**
148      * Add a rule for row of line draw.
149      * @param group is the array.
150      */
151     public void addConnectionGroups(int groups[]) {
152        if (connectionGroups == null) {
153           connectionGroups = new ArrayList<ArrayList>();
154        }
155
156        ArrayList<Integer> row = new ArrayList<Integer>();
157        for(int i = 0; i < groups.length ; i++) {
158           row.add(new Integer(groups[i]));
```

```
159         }
160         connectionGroups.add(row);
161     }
162 
163     /**
164      * Returns a rule of list that contains row of line draw.
165      * @return ArrayList of the line's row
166      */
167     public ArrayList<ArrayList> getConnectionGroups() { return
            connectionGroups; }
168 
169     /**
170      * Sets a rule of list that contains row of line draw.
171      * @param row ArrayList of colors
172      */
173     public void setConnectionGroupsColor(ArrayList<Color> row) {
            connectionGroupsColor = row; }
174 
175     /**
176      * Sets a rule of list that contains row of line draw.
177      * @param row ArrayList of colors
178      */
179     public void setConnectionGroupsColor(Color colors[]) {
180         ArrayList<Color> cl = new ArrayList<Color>();
181         for(int i = 0; i < colors.length; i++) {
182             cl.add(colors[i]);
183         }
184         setConnectionGroupsColor(cl);
185     }
186 
187     public void setPointColor(ArrayList<Color> row) {
188             PointColor = row;
189         }
190 
191     /**
192      * Sets a rule of list that contains row of line draw.
193      * @param row ArrayList of colors
194      */
195     public void setPointColor(Color colors[]) {
196         ArrayList<Color> cl = new ArrayList<Color>();
197         for(int i = 0; i < colors.length; i++) {
198             cl.add(colors[i]);
199         }
200         setPointColor(cl);
201     }
202 
203     /**
204      * Sets a rule of list that contains row of line draw.
205      * @param row ArrayList of colors
```

```
206      */
207     public void setPointColor(int colors[]) {
208        setPointColor(generateColors(colors));
209     }
210
211     /**
212      * Returns a rule of list that contains row of line draw.
213      * @param return ArrayList of colors
214      */
215     public ArrayList<Color> getPointColors() { return PointColor; }
216
217     /**
218      * Sets a rule of list that contains row of line draw.
219      * @param row ArrayList of colors
220      */
221     public void setPointSize(int pointsize[]) {
222        ArrayList<Integer> sizes = new ArrayList<Integer>();
223        for(int i = 0; i < pointsize.length; i++) {
224           sizes.add(new Integer(pointsize[i]));
225        }
226        setPointSize(sizes);
227     }
228
229        public void setPointSize(ArrayList<Integer> row2) {
230           PointSize = row2;
231        }
232
233     /**
234      * Returns a rule of list that contains row of line draw.
235      * @param return ArrayList of colors
236      */
237     public ArrayList<Integer> getPointSize() {
238        return PointSize;
239     }
240
241     /**
242      * Returns a rule of list that contains row of line draw.
243      * @param return ArrayList of colors
244      */
245     public ArrayList<Color> getConnectionGroupsColor() { return
           connectionGroupsColor; }
246
247     /**
248      * Sets a rule of list that contains row of line draw.
249      * @param row ArrayList of Symbols
250      */
251
252     public void setConnectionGroupsSymbol(int[] symbols){
253        setConnectionGroupsSymbol(generateSymbols(symbols));
```

```
254     }
255
256     public void setConnectionGroupsSymbol(ArrayList<Symbol> row) {
257        connectionGroupsSymbol = row;
258     }
259
260   [...]
```

Listing 9.11: Excerpt of `XploRePlotModel.java`

```
1  package jp.jasp.jasplot;
2
3  import java.awt.AlphaComposite;
4  import java.awt.BasicStroke;
5  import java.awt.Color;
6  import java.awt.Graphics2D;
7  import java.awt.Rectangle;
8  import java.awt.Stroke;
9  import java.awt.event.MouseEvent;
10 import java.awt.geom.Line2D;
11 import java.awt.geom.Path2D;
12 import java.util.ArrayList;
13 import java.util.Arrays;
14 import javax.swing.JComponent;
15 import jp.jasp.jasplot.util.symbol.Symbol;
16
17 import java.awt.*;
18 import javax.swing.JComponent;
19 import java.awt.Image;
20 import java.awt.image.BufferedImage;
21 import java.io.*;
22 import javax.imageio.ImageIO;
23 /**
24  * XploRePlotter is a class for drawing the XploRePlotModel.
25  * @author Uwe Ziegenhagen
26  *
27  */
28 public class XploRePlotter extends BasicPlotter {
29    Line2D line = new Line2D.Double();
30    static final long serialVersionUID = -6722837987463154408L;
31    int[][] lines = new int
          [][]{{1,2},{2,3},{3,4},{4,5},{5,6},{6,7},{7,8}};
32    int[] lineWidth = new int[]{1,2,3,4,5,6,7};
33    int[] lineStyle = new int[]{1,2,3,4,5,6,7};
34    int[] lineColor = new int[]{0,1,2,3,4,5,6};
35
36          Image image;
37       int width = -1;
38       int height = -1;
```

```
39
40    /** returns a BasicStroke Object with certain lineWidth and
          linestyle
41     * the linestyle setting for dashed lines is based on the
          thickness of the
42     * line, this algorithm might need some improvement
43     */
44
45    public static BasicStroke getStroke(int lineWidth, int lineStyle){
46       if (lineStyle == 1) {
47          return new BasicStroke(lineWidth, BasicStroke.CAP_SQUARE,
               BasicStroke.JOIN_ROUND, 5.0f, new float[]{1.0f}, 0.0f);
48       } else {
49        Integer i = new Integer(lineWidth);
50        return new BasicStroke(lineWidth, BasicStroke.CAP_SQUARE,
             BasicStroke.JOIN_ROUND, 5.0f, new float[]{i.floatValue(),
             lineStyle*i.floatValue()}, 0.0f);
51       }
52    }
53
54    /** generate ArrayList based on int[] with colors
55     *
56     * @param colors
57     * @return ArrayList
58     */
59    public static java.util.ArrayList generateColors(int colors[]){
60       java.util.ArrayList al = new java.util.ArrayList();
61       for (int i = 0; i<colors.length;i++){
62          switch (colors[i]){
63             case 0: al.add(Color.BLACK);
64             break;
65             case 1: al.add(Color.BLUE);
66             break;
67             case 2: al.add(Color.GREEN);
68             break;
69             case 3: al.add(Color.CYAN);
70             break;
71             case 4: al.add(Color.RED);
72             break;
73             case 5: al.add(Color.MAGENTA);
74             break;
75             case 6: al.add(Color.YELLOW);
76             break;
77             case 7: al.add(Color.WHITE);
78             break;
79             default: al.add(Color.BLACK);
80          }
81       }
82       return al;
```

```
83      }
84      /** generate ArrayList based on int[] with colors
85       *
86       * @param colors
87       * @return ArrayList
88       */
89      public static Color generateColor(int color){
90            switch (color){
91               case 0: return Color.BLACK;
92               case 1: return Color.BLUE;
93               case 2: return Color.GREEN;
94               case 3: return Color.CYAN;
95               case 4: return Color.RED;
96               case 5: return Color.MAGENTA;
97               case 6: return Color.YELLOW;
98               case 7: return Color.WHITE;
99               default: return Color.BLACK;
100         }
101     }
102
103
104     /**
105      * Describe <code>getPlotModel</code> method here.
106      *
107      * @return a <code>XploRePlotModel</code> value
108      */
109     public XploRePlotModel getPlotModel() {
110        return (XploRePlotModel) super.getPlotModel();
111     }
112
113     /**
114      * Creates new XploRePlotter object.
115      * This method is automatically called at the time of
116      * XploRePlotModel creation.
117      * @param model XploRePlotModel
118      */
119     public XploRePlotter(XploRePlotModel model) {
120        setModel(model);
121     }
122
123     /**
124      * Creates a new <code>XploRePlotter</code> instance.
125      *
126      */
127     public XploRePlotter() { }
128
129     int[] selectedObservation = null;
130     /**
131      * Draw a xplore plot
```

```
132      * @param g2 the graphics context
133      * @param compo the JasplotPanel object
134      * @param selectedObservation the selected observation
135      */
136     public void drawData(Graphics2D g2, JasplotPanelPaletteLayer compo
             , int[] selectedObservation) {
137        this.selectedObservation = selectedObservation;
138        drawData(g2, compo);
139        this.selectedObservation = null;
140     }
141
142     /**
143      * Draw a xplore plot
144      * @param g2 the graphics context
145      * @param compo the JasplotPanel object
146      */
147     public void drawData(Graphics2D g2, JasplotPanelPaletteLayer compo
             ) {
148        // The flag of mouse selection
149        // save original stroke
150        Stroke s = g2.getStroke();
151
152        if (getPlotModel().isConnect()) {
153           ArrayList<ArrayList> connectionGroups = getPlotModel().
                  getConnectionGroups();
154           if (connectionGroups == null) {
155              subPlot(g2, compo);
156           } else {
157              subPlotRow(g2, compo, connectionGroups);
158           }
159        } else {
160           plot0(g2, compo);
161        }
162
163        g2.setStroke(s);
164     }
165
166     /**
167      * Draws each data object.
168      */
169     private void plot0(Graphics2D g2, JComponent component) {
170         /* putting a picture in the background
171          try {
172           File file = new File("uwe.png");
173           BufferedImage bImage = ImageIO.read(file);
174           this.image = bImage;
175           width = image.getWidth(null);
176           height = image.getHeight(null);
177           // setPreferredSize (new Dimension (width,height));
```

```
178         }
179           catch (Exception e){
180              System.err.println(e.toString());
181           }
182         */
183
184         g2.drawImage(image,0,0,null);
185
186         if (selectedObservation == null) {
187            g2.setColor(getPlotModel().getUnselectedColor());
188            g2.setComposite(AlphaComposite.getInstance(AlphaComposite.
                   SRC_OVER, getPlotModel().getSelectedCompositeRule()));
189            for (int i = 0; i < getPlotModel().getObservationNumber(); i
                   ++) {
190               eachPlot0(g2, component, i);
191            }
192         } else {
193            g2.setColor(getPlotModel().getSelectedColor());
194            g2.setComposite(AlphaComposite.getInstance(AlphaComposite.
                   SRC_OVER, getPlotModel().getSelectedCompositeRule()));
195
196            for (int i = 0; i < selectedObservation.length; i++) {
197               eachPlot0(g2, component, selectedObservation[i]);
198             }
199         }
200            /* painting of lines
201             * we save the current stroke and the color
202             * since line painting uses its own stroke and color;
203             */
204
205            Stroke backupStroke = g2.getStroke();
206            Color backupColor = g2.getColor();
207
208            /* for each entry in lines there should be an entry
209             * in lineWidth, linestyle and lineColor
210             */
211
212            for (int i = 0; i < lines.length; i++) {
213               g2.setStroke(getStroke(lineWidth[i],lineStyle[i]));
214               g2.setColor(generateColor(lineColor[i]));
215               eachLinePlot0(g2, component, lines[i][0]-1,lines[i
                      ][1]-1);
216            }
217
218
219            g2.setStroke(backupStroke);
220            g2.setColor(backupColor);
221
222         g2.setComposite(AlphaComposite.getInstance(AlphaComposite.
```

```
            SRC_OVER, getPlotModel().getUnselectedCompositeRule()));
223
224         int xColumn = getPlotModel().getXColumn();
225         int yColumn = getPlotModel().getYColumn();
226         DataModel dataModel = getPlotModel().getDataModel();
227         if (getPlotModel().isDrawXZeroLine()) {
228            line.setLine(scaleX(0),
229                     scaleY(dataModel.getMin(yColumn)),
230                     scaleX(0),
231                     scaleY(dataModel.getMax(yColumn)));
232            g2.draw(line);
233         }
234         if (getPlotModel().isDrawYZeroLine()) {
235            line.setLine(scaleX(dataModel.getMin(xColumn)),
236                     scaleY(0),
237                     scaleX(dataModel.getMax(xColumn)),
238                     scaleY(0));
239            g2.draw(line);
240         }
241      }
242
243   private void eachLinePlot0(Graphics2D g2, JComponent component, int i
       ,int j) {
244
245      double xValue1 = getPlotModel().getReal(i, getPlotModel().
          getXColumn());
246      double yValue1 = getPlotModel().getReal(i, getPlotModel().
          getYColumn());
247      double xValue2 = getPlotModel().getReal(j, getPlotModel().
          getXColumn());
248      double yValue2 = getPlotModel().getReal(j, getPlotModel().
          getYColumn());
249
250      double xLocation1 = scaleX(xValue1);
251      double xLocation2 = scaleX(xValue2);
252      double yLocation1 = scaleY(yValue1);
253      double yLocation2 = scaleY(yValue2);
254
255      line.setLine(xLocation1, yLocation1, xLocation2,yLocation2);
256      g2.draw(line);
257   }
258
259      private void eachPlot0(Graphics2D g2, JComponent component, int i)
           {
260         double xValue = getPlotModel().getReal(i, getPlotModel().
             getXColumn());
261         double yValue = getPlotModel().getReal(i, getPlotModel().
             getYColumn());
262         ArrayList<Color> pcolors = getPlotModel().getPointColors();
```

```
263         ArrayList<Integer> psize = getPlotModel().getPointSize();
264         ArrayList<Symbol> symbolsList = getPlotModel().
                getConnectionGroupsSymbol();
265
266         // g2.setBackground(Color.WHITE);
267         Color saveColor = g2.getColor();
268
269         double xLocation = scaleX(xValue);
270         double yLocation = scaleY(yValue);
271
272         // plot symbol
273         Symbol symbol = getPlotModel().getSymbol(i);
274
275         if (symbolsList!= null){
276           symbol = symbolsList.get(i);
277         }
278
279         if (psize!= null){
280            symbol.setSize(psize.get(i).intValue());
281         }
282
283         if (pcolors!= null){
284            g2.setColor(pcolors.get(i));
285         }
286
287         symbol.paint(this, g2, xLocation, yLocation);
288         g2.setColor(saveColor);
289     }
290
291
292
293     /**
294      * Draws a main title, ticks of coordinate, title of coordinate.
295      * @param g2 g2 the graphics context
296      * @param compo the editor component
297      */
298     public void drawRulers(Graphics2D g2, JasplotPanelPaletteLayer
            compo) {
299         drawTitle(g2);
300
301         // draw ticks
302         if(getPlotModel().isDrawXTick() || getPlotModel().isDrawYTick()
                ) {
303            plotTicks(g2);
304         }
305
306         // X label
307         String xAxisLabel = getPlotModel().getXAxisLabel();
308         if(getPlotModel().isDrawXAxisLabel() && xAxisLabel.length() >
```

```
            0) {
309         drawXVariableName(g2, compo, getPlotModel().getXAxisLabel())
                ;
310      }
311      // Y label
312      if(getPlotModel().isDrawYAxisLabel()) {
313         drawYVariableName(g2, compo, getPlotModel().getYAxisLabel())
                ;
314      }
315   }
316
317   /**
318    * The mode which draws the line of one usual data.
319    * @param g2 the graphics context
320    * @param compo the editor component
321    */
322   private void subPlot(Graphics2D g2, JComponent compo) {
323      double xLocation0 = scaleX(getPlotModel().getReal(0,
          getPlotModel().getXColumn()));
324      double yLocation0 = scaleY(getPlotModel().getReal(0,
          getPlotModel().getYColumn()));
325      System.out.println("hobo1");
326      int size = getPlotModel().getObservationNumber();
327
328      for (int i = 1; i < size ; i++) {
329         double xLocation = scaleX(getPlotModel().getReal(i,
             getPlotModel().getXColumn()));
330         double yLocation = scaleY(getPlotModel().getReal(i,
             getPlotModel().getYColumn()));
331
332         line.setLine(xLocation0, yLocation0, xLocation, yLocation);
333         g2.setColor(getPlotModel().getSymbolColor(i));
334         g2.draw(line);
335
336         // set next line;
337         xLocation0 = xLocation;
338         yLocation0 = yLocation;
339      }
340
341   }
342
343   /**
344    * The mode which is specified sequence.
345    * @param g2 the graphics context
346    * @param compo the editor component
347    * @param al the ArrayList of line's row
348    */
349   private void subPlotRow(Graphics2D g2, JComponent compo, ArrayList
          <ArrayList> connectionGroups) {
```

```
350         ArrayList<Color> colors = getPlotModel().
                getConnectionGroupsColor();
351        // ArrayList<Color> pcolors = getPlotModel().getPointColors();
352         ArrayList<Symbol> symbols = getPlotModel().
                getConnectionGroupsSymbol();
353         ArrayList<Double> thicknesses = getPlotModel().
                getConnectionGroupsThickness();
354         ArrayList<float []> stylelist = getPlotModel().
                getConnectionGroupsStyle();
355
356         PlotModelHandler plotModelHandler = ((JasplotPanelPaletteLayer)
                compo).getPlotModelHandler();
357
358          for (int i = 0; i < connectionGroups.size(); i++) {
359            ArrayList connectionGroup = connectionGroups.get(i);
360            Integer idx = (Integer) connectionGroup.get(0);
361            int index = idx.intValue();
362            double xLocation0 = scaleX(getPlotModel().getReal(index,
                   getPlotModel().getXColumn()));
363            double yLocation0 = scaleY(getPlotModel().getReal(index,
                   getPlotModel().getYColumn()));
364
365            // plot a symbol for line's each end-mark
366            Symbol symbol;
367
368            if (symbols != null) {
369               symbol = symbols.get(i);
370            } else {
371               symbol = getPlotModel().getSymbol(index);
372            }
373
374            symbol.paint(this, g2, xLocation0, yLocation0);
375
376             // g2.setColor(colors.get(i));
377
378            // color
379            if (colors != null) {
380               g2.setColor(colors.get(i));
381            }
382
383            int colorIndex = 0;
384             for (int j = 0; j < connectionGroup.size(); j++) {
385                idx = (Integer) connectionGroup.get(j);
386                index = idx.intValue();
387               if (plotModelHandler.isSelectedObservation(index)) {
388                  colorIndex = index;
389                  break;
390               }
391            }
```

```
392
393             if (colorIndex != 0) {
394                g2.setColor(getPlotModel().getIndexColor());
395             } else {
396                if (colors != null) {
397                   g2.setColor(colors.get(i));
398                }
399             }
400
401             // storke
402             BasicStroke bs = null;
403             Double d = new Double(1.0);
404             if (thicknesses != null) {
405                d = thicknesses.get(i);
406             }
407
408             try {
409                bs = new BasicStroke(d.floatValue(), BasicStroke.CAP_BUTT
                      , BasicStroke.JOIN_MITER, 10.0f, stylelist.get(i), 0.0
                      f);
410             } catch (NullPointerException e) {
411                bs = new BasicStroke(d.floatValue(), BasicStroke.CAP_BUTT
                      , BasicStroke.JOIN_MITER, 10.0f, null, 0.0f);
412             }
413
414             // plot lines
415             for (int j = 0; j < connectionGroup.size(); j++) {
416                idx = (Integer) connectionGroup.get(j);
417                index = idx.intValue();
418
419                double xLocation = scaleX(getPlotModel().getReal(index,
                      getPlotModel().getXColumn()));
420                double yLocation = scaleY(getPlotModel().getReal(index,
                      getPlotModel().getYColumn()));
421
422                if (bs != null) {
423                   g2.setStroke(bs);
424                }
425
426                line.setLine(xLocation0, yLocation0, xLocation, yLocation
                      );
427                g2.draw(line);
428
429                symbol.paint(this, g2, xLocation, yLocation);
430
431                xLocation0 = xLocation;
432                yLocation0 = yLocation;
433             }
434          }
```

```
435
436     }
437
438     public void showPopupMenu(MouseEvent e, JComponent c) {
439        System.out.println("XploRePlotter.showPopupMenu");
440     }
441
442     /**
443      * Describe <code>getIndexForLocation</code> method here.
444      *
445      * @param region a <code>Rectangle</code> value
446      * @param indexes a <code>boolean[]</code> value
447      * @return a <code>boolean[]</code> value
448      */
449     public boolean[] getIndexForLocation(Path2D path2d, boolean[]
           indices, JComponent c) {
450 //       Arrays.fill(indices, false);
451
452        Symbol symbol;
453
454        for (int i = 0; i < getPlotModel().getObservationNumber(); i
              ++) {
455           symbol = getPlotModel().getSymbol(i);
456           if (symbol.intersects(path2d)) {
457              indices[i] = true;
458           }
459        }
460        return indices;
461     }
462 }
```

Listing 9.12: `XploRePlotter.java`

The following figures 9.10 to 9.16 show various plots generated by using the `XploRePlotModel` and show how Jasplot is able to handle XploRe graphical objects. The underlying calculations were made in XploRe as Jasp cannot handle XploRe syntax, yet.

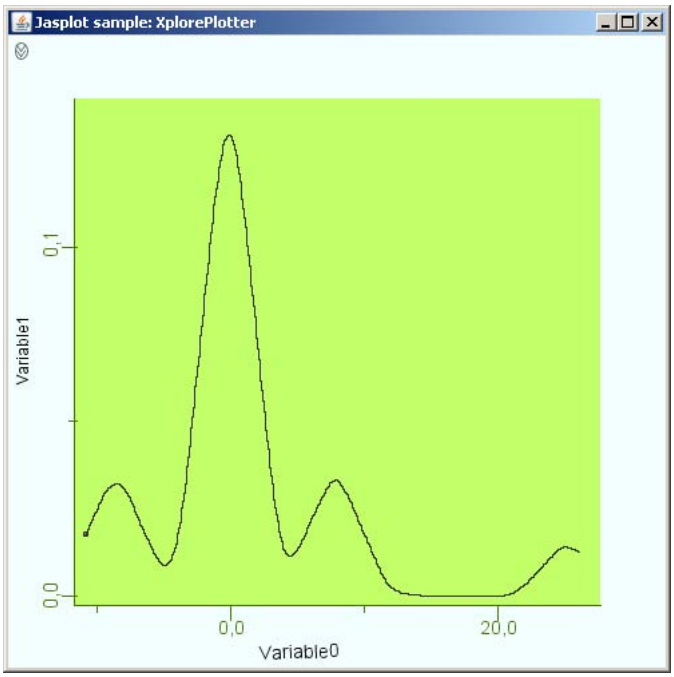


Abbildung 9.10: Average shifted histogram

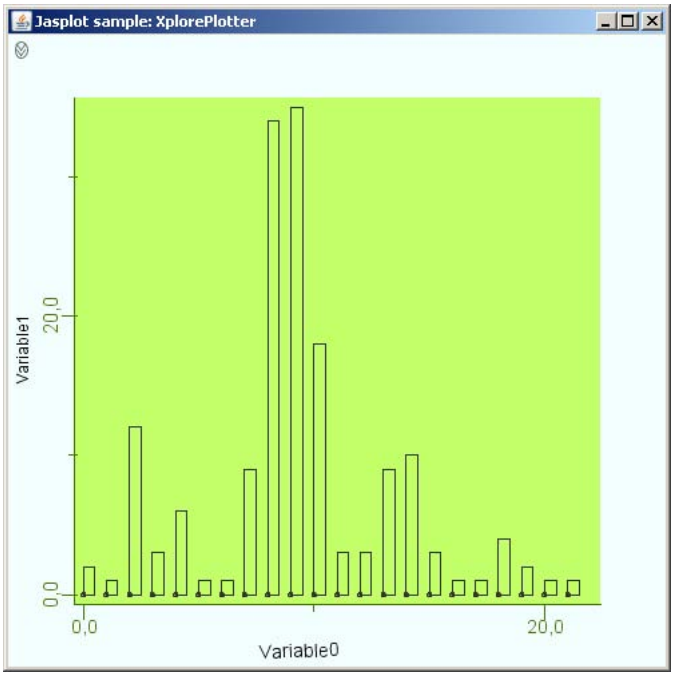


Abbildung 9.11: Barchart

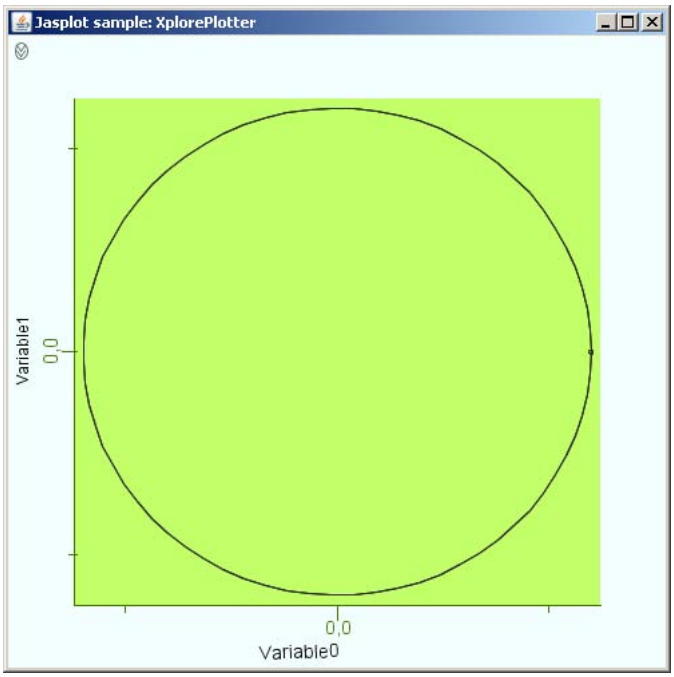


Abbildung 9.12: Graphical primitive: circle

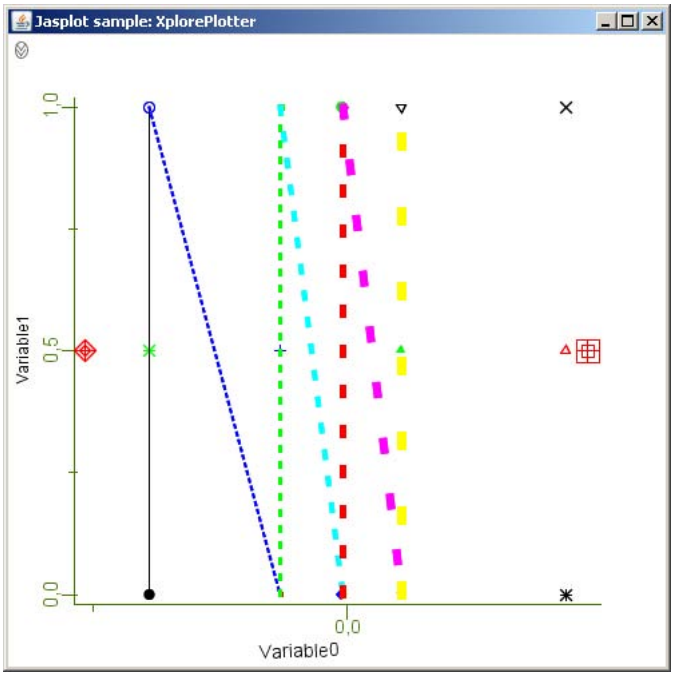


Abbildung 9.13: Different line styles and colors

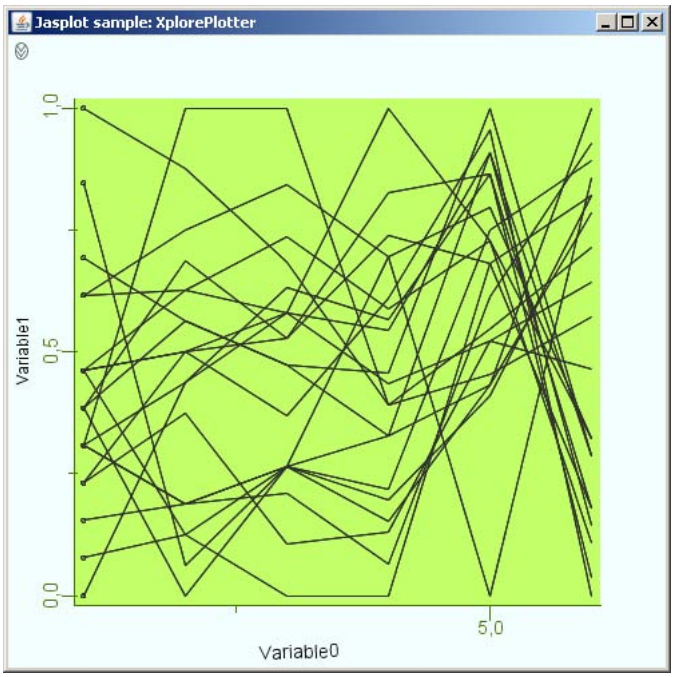


Abbildung 9.14: Parallel coordinate plot

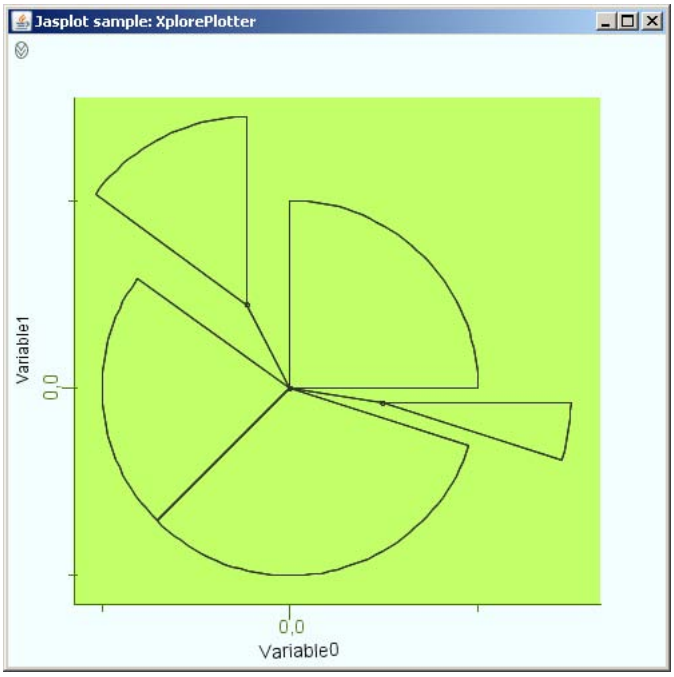


Abbildung 9.15: Pie chart

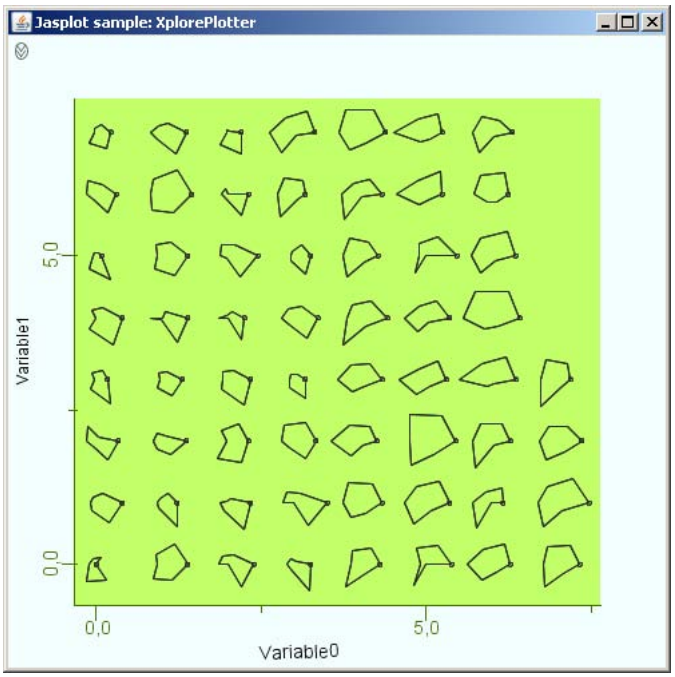


Abbildung 9.16: Star plot

9.6 Application to R

In recent years the statistical programming language **R** received a lot of attention from both, statisticians and developers, who made this software a lingua franca in statistical applications for all sciences. Various user interfaces are available for **R**, from standalone applications to embedded versions and plugins for editors such Emacs. With JGR (Helbig et al., 2005) there is even a highly sophisticated Java client available, offering far more functionality than the desktop version of **R**.

Nevertheless there are applications where a fully featured client is not needed, for example when small code samples are to be embedded into webpages. With the Java applet technology fully-featured programs can be embedded which connect to computing and databases servers and show the result to the user. With the combination of parts of the Yxilon Java Client with applet technology and the ability to use **R** as computing engine we allow, just like (Borak et al., 2005) with the XploRe Quantlet Client (XQC) the education of students with realworld examples.

The client downloads **R** code, which has been specified in the `applet` tag of the HTML code of the webpage and executes it. Results are shown in a small output window respectively a plot window.

9.6.1 R Server and communication protocol

The connection with **R** is made, just as in Yxilon, using a proprietary protocol on the basis of TCP/IP. Listing 9.13 shows a very basic example of the communication between Java and **R**. This code opens a new connection to the `Rserve` executable, running on the same machine, and sends a simple concatenation command of a string vector. The result of this operation, the components of the string vector, are then printed out elementwise.

```
import org.rosuda.REngine.*;
import org.rosuda.REngine.Rserve.*;

public class BasicTest {
   public static void main(String[] args) {
   try {
     RConnection c = new RConnection();
     REXP r = c.eval("c('a','b','c')");
     String[] s = r.asStrings();
     for (int i=0;i<s.length;i++){
       System.out.println(s[i]);
     }
   } catch (Exception e) {
      e.printStackTrace();
     }
   }
}
```

Listing 9.13: `BasicTest.java`, a minimal example for Rserve

As a `REXP` object is only a container for different matrix, list or vector objects, a fully-featured client has to distinguish carefully between the different object types. Listing 9.14 shows the respective `evaluate()` function of the RJ-Client.

```
public void evaluate(REXP rx){
  output.setText(output.getText()+ " String " + rx.isString()+"\n");
  output.setText(output.getText()+ " Numeric " + rx.isNumeric());
  output.setText(output.getText()+ " Integer " + rx.isInteger()+"\n");
  output.setText(output.getText()+ " Null " + rx.isNull());
  output.setText(output.getText()+ " Factor " + rx.isFactor()+"\n");
  output.setText(output.getText()+ " List " + rx.isList());
  output.setText(output.getText()+ " Logical " + rx.isLogical()+"\n");
  output.setText(output.getText()+ " Env " + rx.isEnvironment());
  output.setText(output.getText()+ " Lang " + rx.isLanguage()+"\n");
  output.setText(output.getText()+ " Expr " + rx.isExpression());
  output.setText(output.getText()+ " Symbol " + rx.isSymbol()+"\n");
  output.setText(output.getText()+ " Vector " + rx.isVector());
  output.setText(output.getText()+ " Raw " + rx.isRaw()+"\n");
  output.setText(output.getText()+ " Complex " + rx.isComplex());
  output.setText(output.getText()+ " Recur " + rx.isRecursive()+"\n");
```

```
17
18  try {
19    if (rx.isString()==true){
20      if (rx.isVector()==true){ // all strings are vectors
21        String[] s = rx.asStrings();
22        output.setText(output.getText() + "[1] ");
23        for (int i=0;i<s.length;i++){
24          output.setText(output.getText() + s[i] + " ");
25        }
26      } else {
27           output.setText(output.getText() + "\n" + rx.asString());
28      }
29    } // end of if string
30
31    if (rx.isRaw()==true){
32      Image img = Toolkit.getDefaultToolkit().createImage(rx.asBytes())
          ;
33      final Frame f = new Frame("Graphical Output");
34      f.add(new PlotDemo(img));
35      f.addWindowListener(new WindowAdapter() { // just so we can close
            the window
36      public void windowClosing(WindowEvent e) {
37        f.setVisible(false);
38      }
39      });
40      f.pack();
41      f.setVisible(true);
42    }
43
44    if (rx.isList()==true){
45      RList rl = rx.asList();
46      String[] keys = rl.keys();
47      if (keys != null){
48        System.out.println("keys " + keys.length);
49        for (int l=0;l<keys.length;l++){
50          System.out.println(keys[l]);
51        }
52      }
53      int size = rl.size();
54      System.out.println("Size of the list " + size + "\n");
55      for (int k =0;k<size;k++){
56        REXP t = rl.at(k);
57        evaluate(t);
58      }
59      }
60
61      if (rx.isNumeric()==true){
62        if (rx.isVector()==true){
63          if (rx.isInteger()==true){
```

```
64            int[] t = rx.asIntegers();
65            output.setText(output.getText() + "[1] ");
66            for (int i=0;i<t.length;i++){
67            output.setText(output.getText() + t[i] + " ");
68            } // end of int[] handling
69           }else {
70            double[] d = rx.asDoubles();
71            output.setText(output.getText() + "[1] ");
72            for (int i=0;i<d.length;i++){
73              output.setText(output.getText() + d[i] + " ");
74            }
75           } // end of double[] handling
76         } else {
77            double d = rx.asDouble();
78            output.setText(output.getText() + "\n[1]" + d + "\n ");
79         }
80      }
81      } catch (Exception e){
82         output.setText(e.toString());
83      }
84      output.setText(output.getText()+"\n\n");
85  }
```

Listing 9.14: `evaluate()` function from `RJ-Client.java`

Special focus may be given to the way graphics are handled in this application. **R** offers a variety of different graphics devices for Postscript, PDF, etc.; with JavaGD (`http://www.rforge.net/JavaGD/`) there is also a Java-based device which can display graphics created via rJava (`http://stats.math.uni-augsburg.de/rJava/`), a **R**-connector for Java software as it is e.g. used in JGR.

For Rserve we use an alternative suggested by the Rserve developers. Graphics files are created on the server and sent in binary form to the client to be displayed. A first result, depicting charts from the upcoming book by S. Klinke, is shown in the Figures 9.17 and 9.18.

Further work will include the setup of an Rserve instance, the adjustment of the necessary Yxilon classes and, given the technical options, the inclusion into Quantnet.

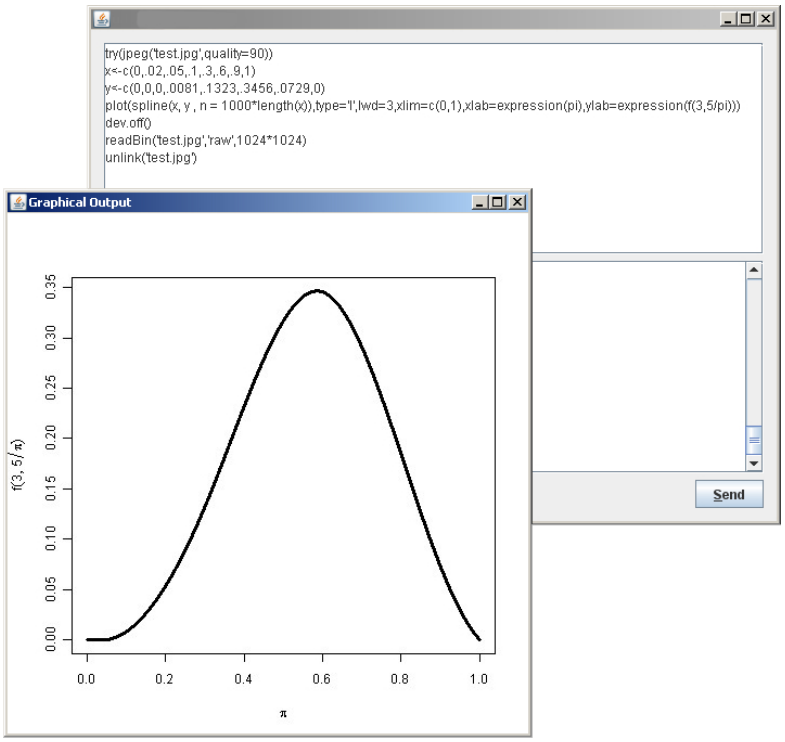


Abbildung 9.17: Graphics output with binary image transmission

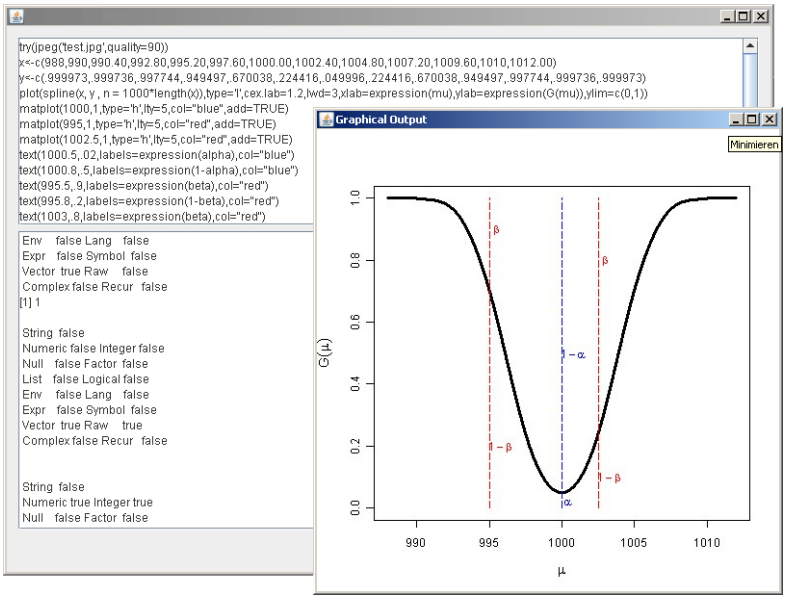


Abbildung 9.18: Graphics output with binary image transmission

Index

Literaturverzeichnis

Taleb Ahmad, Wolfgang Härdle, and Sigbert Klinke. Using wiki to build an e-learning system in statistics in arabic language. In CESSE, editor, *CESSE 2007 Proceedings of the XXI. International Conference on Computer, Electrical, and Systems Science, and Engineering, Vienna*, pages 338–342, 2007.

G. Aydınlı, W. Härdle, and E. Neuwirth. Efficient and Secure Statistics in Office Applications. In M. Minnotte, J. Symanzik, and E. Wegman, editors, *Proc. of the 35th Symposium on the Interface "Security and Infrastructure Protection"*, 2003.

Bitkom. Pressemitteilung: Zwei drittel arbeiten mit computern. Software, 2008. URL `http://www.bitkom.org/de/presse/8477_49790.aspx`. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.

Szymon Borak, Wolfgang Härdle, and Heiko Lehmann. *Statistical Tools for Finance and Insurance*, chapter Working with the XQC. Springer, 2005.

Ulrike Brandes. Statistische Bewertung und Analyse der Klausurergebnisse Statistik (Grundstudium). Diploma thesis, 2004. URL `http://edoc.hu-berlin.de/docviews/abstract.php?lang=ger&id=26854`.

John Chambers and Duncan Temple Lang. $\widehat{\Omega}$ – a component-based statistical computing environment. In *Proceedings of the 52nd Session of the ISI, Helsinki, Finland.*, 1999. URL `http://cm.bell-labs.com/stat/doc/ISI99Omegahat.pdf`.

John M. Chambers. Users, programmers, and statistical software. *ASA Journal of Comp. and Graph. Stat.*, 9:3:404–422, 2000. URL `http://cm.bell-labs.com/stat/doc/jmcJCGS2000.ps`.

H. Chernoff. The use of faces to represent points in *k*-dimensional space graphically. *Journal of the American Statistical Association*, 68(342), 1973.

E. Cramer, C. Kramer, and U. Kamps. e-stat: A web-based learning environment in applied statistics. In *Proceedings in Computational Statistics - 15th Symposium held in Berlin, Physika Verlag, Heidelberg*, pages 309–315, 2002. URL `http://emilea-stat.rwth-aachen.de`.

Paul L. Darius, Kenneth M. Portier, and Eddie Schrevens. Data collection skills: Challenges for teaching and training. *International Statistical Review*, 2007.

Christine Duller. A kind of PISA-survey at university. In J. Antoch, editor, *Proceedings in Computational Statistics - 16th Symposium held in Prague, Physika Verlag, Heidelberg*, pages 975–980, 2004.

Christine Duller. Do you speak statistics? In n.n., editor, *Proceedings of the ISI*, 2007a.

Christine Duller. Doing statistics versus understanding statistics. In *DAGStat - Statistics under one Umbrella*, page 88, 2007b.

W. J. Eckert and J.C. McPherson. *Punched Card Methods in Scientific Computation (Charles Babbage Institute Reprint)*. The MIT Press, 1984. ISBN 0262050307.

Jörg Feuerhake. Optimierung client-/serverbasierter Statistiksysteme. Diploma thesis, 2002. URL `http://www.edoc.hu-berlin.de/docviews/abstract.php?lang=ger&id=26913`.

B. Flury and H. Riedwyl. Graphical representation of multivariate data by means of asymmetrical faces. *Journal of the American Statistical Association*, 76(376), 1981.

Elisabeth Freeman, Eric Freeman, Bert Bates, and Kathy Sierra. *Head First Design Patterns (Head First)*. O'Reilly Media, Inc., 2004. ISBN 0596007124.

Takeshi Fujiwara, Uwe Ziegenhagen, Yoshikazu Yamamoto, Junji Nakano, and Wolfgang Härdle. Using statistical libraries in different statistical systems. *IASC Proceedings (submitted)*, December 2008.

Tomokazu Fujiwara, Kobayashi Ikunori, Junji Nakano, and Yoshikazu Yamamoto. A statistical package based on pnuts. In *COMPSTAT. Proceedings in Computational Statistics*, 2000.

C. Grune. Neue statistik. Talk at Learntec 2002, Karlsruhe, 2002.

Yuval Guri, Sigbert Klinke, and Uwe Ziegenhagen. Yxilon – a modular open-source statistical programming language. In *Proceedings of the 55th Session of the International Statistical Institute, Sydney, Australia*, 2005. URL `http://sfb649.wiwi.hu-berlin.de/papers/pdf/SFB649DP2005-018.pdf`.

Markus Helbig, Martin Theus, and Simon Urbanek. JGR: Java GUI for **R**. *Statistical Computing & Graphics*, 16(2):9–12, December 2005. URL `http://www.statcomputing.org/newsletter/v162.pdf`.

W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer, 2003.

W. Härdle, S. Klinke, and M. Müller. *XploRe Learning Guide*. Springer, 2000.

W. Härdle, J. Franke, and C. Hafner. *Einführung in die Statistik der Finanzmärkte*. Springer, 2nd edition, 2004a.

Wolfgang Härdle, Sigbert Klinke, and Uwe Ziegenhagen. Yxilon - designing the next generation, vertically integrable statistical software environment. In *Proceedings of the 36th Syposium on the Interface, Baltimore, USA*, 2004b. URL `http://lehre.wiwi.hu-berlin.de/Professuren/quantitativ/statistik/members/personalpages/uz/publications/haeklizie-yxilon-040802.pdf`.

Michael Kerres. http://mediendidaktik.uni-duisburg-essen.de/leitbild. Mission statement, 2005. URL `http://mediendidaktik.uni-duisburg-essen.de/leitbild`.

Keypress. Fathom 2. Software, 2007. URL `http://www.keypress.com/x5656.xml`.

S. Klinke and H. Lehmann. MD*Book and XQC – an architecture for reproducible research. SFB 373 Research Paper, 2003. URL `http://sfb.wiwi.hu-berlin.de`.

S. Klinke and R. Witzel. MD*Book online – a tool for creating interactive documents. SFB 373 Research Paper, 2002. URL `http://sfb.wiwi.hu-berlin.de`.

Sigbert Klinke. Q&A - variable multiple choice exercises with commented answers. In J. Antoch, editor, *Proceedings in Computational Statistics - 16th Symposium held in Prague, Physika Verlag, Heidelberg*, pages 1299–1304, 2004.

Sigbert Klinke and Olga Zlatkin-Troitschanskaia. Embedding R in the Mediawiki. SFB 649 Discussion Paper 2007-061, Sonderforschungsbereich 649, Humboldt Universität zu Berlin, Germany, 2007. available at http://sfb649.wiwi.hu-berlin.de/papers/pdf/SFB649DP2007-061.pdf.

R. Kristöfl. Evaluation von lernplattformen: Verfahren, ergebnisse und empfehlungen, report of the austrian ministry for education, science and culture. online, 2005. URL `http://moodle.de/mod/resource/view.php?id=706`.

M. Krätzig. Creating user interfaces for econometric routines with JStatCom: An example for Ox. to be presented on the 2nd Oxmetrics User Conference, August 2004, 2004.

H. Lehmann. XploRe Quantlet Client – web service for mathematical and statistical computing. SFB 373 Research Paper, 2003. URL `http://sfb.wiwi.hu-berlin.de`.

H. Lehmann. *Client/Server based statistical computing*. Dissertation, Humboldt-Universität zu Berlin, 2004.

Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

B. McCullough and B. Wilson. On the accuracy of statistical procedures in Microsoft Excel 97. *Computational Statistics & Data Analysis*, 1(31):27–39, 1999.

B. McCullough and B. Wilson. On the accuracy of statistical procedures in Microsoft Excel 2003. *Computational Statistics & Data Analysis*, 49(4):1244–1253, 2005.

McGraw-Hill. Visual statistics. Software, 2001. URL `http://www.mhhe.com/business/opsci/doane/home.htm`.

G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 1956.

Miscellaneous. Wikibook Statistics, chapter 'Numerical Methods'. online, 2004. URL `http://en.wikibooks.org/wiki/Statistics`.

Yuichi Mori, Yodobashi Yamamoto, and Hirohisa Yadohisa. Data-oriented learning system of statistics based on analysis scenario/story (dolstat). *Bulletin of the International Statistical Institute*, 2003.

Marlene Müller, Bernd Rönz, and Uwe Ziegenhagen. The multimedia project MM*STAT for teaching statistics. In J. Bethlehem and P. van der Heijden, editors, *Proceedings in Computational Statistics - 14th Symposium held in Utrecht*, pages 409–415, 2000.

Junji Nakano. Parallel computing in a statistical system jasp. In Jaromir Antoch, editor, *COMPSTAT Proceedings in Computational Statistics*, pages 2003–2010, 2004.

J. Nielsen. *Usability Engineering*. AP Professional, 1993.

J. Nielsen. Usability 101. Jakob Nielsen's Alertbox, 2003. URL `http://www.useit.com/alertbox/20030825.html`.

Deborah Nolan and Duncan Temple Lang. Dynamic, interactive documents for teaching statistical practise. *International Statistical Review*, 2007.

Bernd Rönz and Hans Strohe. *Lexikon Statistik*. Gabler, 1996.

B. Shneiderman. *Designing the User Interface*. Addison-Wesley Longman, 3. edition, 1997.

D. Stock and C. Watson. Human judgement accuracy, multidimensional graphics, and human versus models. *Journal of Accounting Research*, 22(1), 1984.

M. Theus. Java - the next generation of statistical computing? In *Proceedings of the 30th Symposium on the Interface*, 1998.

M. Theus. User interfaces of interactive statistical graphics software. In *Proceedings of the 31th Symposium on the Interface*, 1999.

Tomatsu. The pnuts script language for java. Project Homepage, 2008. URL `http://pnuts.dev.java.net/`.

J. W. Tukey. The technical tools of statistics. *American Statistician*, April 1965.

Antony Unwin, Martin Theus, and Heike Hofmann, editors. *Graphics of Large Datasets: Visualizing a Million*. Springer, 1 edition, 2006.

Wikipedia. 'Design pattern'. online, 2008a. URL `http://en.wikipedia.org/wiki/Design_pattern_(computer_science)`.

Wikipedia. 'Singleton pattern'. online, 2008b. URL `http://en.wikipedia.org/wiki/Singleton_pattern`.

Yoshikazu Yamamoto, Junji Nakano, and Keisuke Honda. A java library for statistical graphs using design patterns. *Proceedings of the Institute of Statistical Mathematics*, 55 (1):27–45, 2007.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 29. September 2008 Uwe Ziegenhagen